Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

И.В. Гаврилова

ПРОГРАММИРОВАНИЕ

*Утверждено Редакционно-издательским советом университета
в качестве практикума*

Магнитогорск
2022

УДК 004.42+37
ББК 74.4

Рецензенты:

кандидат педагогических наук,
учитель информатики,
МОУ «Средняя общеобразовательная школа №28»
г. Магнитогорска
А.С. Доколин

кандидат технических наук,
начальник управления ИТ и АСУ,
ФГБОУ ВО «Магнитогорский государственный технический
университет им. Г.И. Носова»
К.А. Рубан

Гаврилова И.В.
Программирование [Электронный ресурс] : практикум / Ирина Викторовна Гаврилова ; ФГБОУ ВО «Магнитогорский государственный технический университет им. Г.И. Носова». – Электрон. текстовые дан. (1,29 Мб). – Магнитогорск : ФГБОУ ВО «МГТУ им. Г.И. Носова», 2022. – 1 электрон. опт. диск (CD-R). – Систем. требования : IBM PC, любой, более 1 GHz ; 512Мб RAM ; 10 Мб HDD ; MS Windows XP и выше ; Adobe Reader 8.0 и выше ; CD/DVD-ROM дисковод ; мышь. – Загл. с титул. экрана.

Практикум составлен в соответствии с рабочей программой дисциплины «Программирование». Содержит перечень индивидуальных практических заданий по всем разделам дисциплины, примеры программ и ссылки на учебные материалы, необходимые или полезные при решении задач. Несмотря на то, что примеры программ представлены на языке C++, задания не ориентируются на какой-либо конкретный язык программирования и могут быть взяты за основу при постановке практикума по программированию на любом языке.

Практикум предназначен для студентов направлений подготовки 09.03.03 Прикладная информатика.

УДК 004.42+37
ББК 74.4

Содержание

ВВЕДЕНИЕ

Успешное освоение программирования подразумевает активное написание программ, при этом начинать нужно с самых простых, несложных, направленных на отработку отдельных умений: ввод информации с клавиатуры, вывод на экран, использование операторов ветвлений, циклов и т.п. Из этих простых шагов, подчиненных алгоритму, складывается программа или даже комплекс программ, выполняющих сложные задачи.

Прежде чем приступить к программированию, нужно решить, в какой среде будет осуществлять разработка. Практика подсказывает, что для начала сверхсложные среды программирования, например, Visual Studio или QT Creator не нужны, достаточно простых онлайн-компиляторов, которые не требуют установки. Это связано с тем, что «большие» программы зачастую накладывают специфику на процесс разработки, упрощая её для практиков и усложняя для новичков за счёт использования дополнительных библиотек. Примеры онлайн-компиляторов приводятся в конце рекомендаций, они очень похожи друг на друга, отличаются лишь версиями поддерживаемых языков.

Решение задачи с помощью ПК — это процесс автоматического преобразования исходной информации (исходных данных) в искомый результат в соответствии с заданной последовательностью действий, называемой алгоритмом. Алгоритм — это точное предписание, которое задает вычислительный процесс, начинающийся из некоторой совокупности возможных для этого алгоритма исходных данных и направленный на получение полностью определяемого этими исходными данными результата.

Основными этапами подготовки задачи к решению на ПК, которые связаны с построением алгоритма, являются:

1) постановка задачи: построение модели данных и разработка алгоритма;
2) реализация алгоритма в виде программы;
3) проверка правильности алгоритма;
4) анализ сложности алгоритма.

Этап постановки задачи - это основа решения задачи. Для того чтобы правильно понять задачу, следует придерживаться следующих правил.

1. Выделить в задаче исходные данные и требуемые результаты.
2. Определить структуры данных, которые можно использовать для представления исходных данных и результатов в алгоритме.
3. Сформулировать постановку задачи в виде одного короткого, конкретного предложения. Для этого, в исходной постановке задачи необходимо исключить все уточняющие детали, в то же время, зафиксировав их в другом месте. В процессе анализа рекомендуется составить макет исходных данных, макет печати результатов и таблицу идентификаторов.

Макет исходных данных - это форма представления исходных данных с записью конкретных значений. При составлении макета целесообразно сразу же назначать имена объектам и записывать их на макете. Макет печати результатов - это форма выходного документа. В нем должны быть предусмотрены необходимые поясняющие тексты.

“Заготовка” таблицы идентификаторов делается в начале анализа постановки задачи. Она не закрывается до конца решения задачи, так как в любой момент составления алгоритма может быть дополнена. Таблица идентификаторов в дальнейшем служит основой для записи объявлений в алгоритме.

При составлении алгоритма можно использовать метод пошаговой детализации, который заключается в том, что на каждом шаге алгоритм расширяется по мере добавления и уточнения деталей. Уровень команд (инструкций) постепенно понижается. Весь процесс заканчивается тогда, когда достигнут уровень системы команд непосредственного исполнителя алгоритма.

После того, как переменные будут описаны, взаимосвязи между ними установлены, необходимо «перевести задачу на язык программирования». Каждый оператор – это команда компилятору, какие действия нужно выполнить с переменной: вывести, прочитать, присвоить ей значение и т.п.

Готовую программу обязательно нужно проверить! Для этого сначала надо придумать несколько наборов данных и соответствующих им результатов, а затем запустить программу на выполнение и ввести данные из примера. Для понимания того, как обрабатывается переменная рекомендуется выводить промежуточные значения – так проще найти команду, выполнение которой приводит к ошибке. Когда всё заработает верно, команды для вывода данных можно удалить или закомментировать.

В представленном практикуме приводятся задания для самостоятельного решения при изучении курса «Программирование». Для удобства все задачи разбиты на два раздела: структурное программирование и объект-ориентированное программирование. Перед каждым вариантом приводится пример решения подобной задачи, после заданий предлагаются ссылки на учебные материалы.

1. СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ

1.1. Программирование линейных алгоритмов

Задание 1

Составить программу для вычисления значения функции для любого заданного *x*. При выводе исходных данных результат округлять до второго знака после запятой.

Пример реализации

Составить программу для вычисления значения функции для любого заданного x. При выводе исходных данных результат округлять до второго знака после запятой.

$$y = 2^{-x} - \cos x + \frac{\sin(2x)}{\ln\left(\cos\frac{e^{-x}}{x^2+1} + 2\right)}.$$

```
/*Объявление значений с плавающей точкой "x" и "y".
"x" вводится пользователем, а "y" является результатом.*/
double x, y;
int main() {
    //Ввод значения "x" и сопровождающее его сообщение.
    cout << "Enter the x: ";
    cin >> x;
    /*Формула для вычисления "y"..*/
    y = pow(2, !x) - cos(x) + (sin(2 * x) / log(cos(pow(exp(x),
!x) / (pow(x, 2)) + 1) + 2));
    //Вывод результата "y", округлённого до сотых, и
сопровождающее его сообщение.
    cout << "The result is " << round(y * 100) / 100 << endl;
    return 0;
}
```

Варианты

1 $y = x\ln\left(x^4+5\right) + \frac{x}{\cos x + 2} + 2\sin\sqrt{x^2+1}$

2 $y = \frac{\sin x + \cos x}{\sqrt[3]{\cos x - \sin x + 3}} + \left(e^x + 1\right)^2$

3 $y = \frac{\cos x}{\pi + 2x^2} + x\ln\left(x^4 + \sqrt{|x|}\right)$

4 $y = \frac{1+\sin\sqrt{|x+1|}}{\cos^2\left(12x^2+4\right)+1} + e^{\frac{\sqrt{|x|}}{\sin^2 x+2}}$

5 $y = x + \frac{x^3}{3} - \frac{x^5}{5} + \frac{xe^{2x}}{\sin^2 x + 1}$

6 $y = \frac{\ln|\cos x|}{\ln\left(1+x^2\right)} + e^{2^x} + 10^{\sin x}$

7 $$y = 3^x + 4x - \frac{\ln\left|\frac{\cos^2 x + 1}{x^2 + 1}\right|}{2\sin x + 25}$$

8 $$y = 2\ln\left|\frac{\sin x + 2}{\cos^2 x + 1}\right| + e^{\sin^2 x + 1}$$

9 $$y = \left|x^2 - x^3\right| - \frac{7x}{\left(x^3 + 15x\right)^2 + 1} + e^{\cos^2 x + 1}$$

10 $$y = x\ln\left(x^2 + 1\right) + \frac{x}{\cos^2 x + 1} + \sqrt{|x + 1|}$$

11 $$y = \sin\sqrt{x^2 + 1} - \frac{\sin\sqrt{|x - 1|}}{\ln\left(x^2 + 1\right)}$$

12 $$y = e^{-2x+1} + \frac{x^2 + 12x - 3}{\ln\left|\sin^3 x + 1\right|}$$

13 $$y = \frac{1 + \sin\sqrt{x^2 + 1}}{\cos^2 12x + 1} + \ln\left(1 + x^2\right)$$

14 $$y = 2\cos 3x - \frac{\ln|\cos x|}{\ln\left(1 + x^2\right)} + e^{\cos^2 x \sin x}$$

15 $$y = e^{\sin^2 x + 1} - \frac{x}{\ln\left|\cos^3 x + 5\right|} + (1 + ч)^x$$

Задание 2

Составить программу для решения задач, согласно варианту. Для каждой задачи выводить пояснения при вводе данных и полный ответ при выводе, например:

```
Введите длину в милях: 2500
Длина в километрах равна 1.34749
```

Пример реализации

```
/*Дано расстояние между городами в километрах (вводится с
клавиатуры).
Перевести его в версты и вывести на экран.*/

/*Объявление значений с плавающей точкой "x" и "y".
"x" вводится пользователем, а "y" является результатом.*/
double x, y;
int main() {
    //Ввод значения "x" и сопровождающее его сообщение.
    cout << "Enter the distance between two cities in kilometers:
";
    cin >> x;
    //Перевод "x" из километров в верста.
    y = x * 0.937383;
```

```
    //Вывод результата "y" и сопровождающее его сообщение.
    cout << "This is " << y << " in verst" << endl;
    return 0;
    }
```

Варианты

1. Дано десятичное число n (вводится с клавиатуры). Вывести количество сотен.
2. Дан размер файла в битах (вводится с клавиатуры). Вывести размер этого файла в килобайтах.
3. Дан размер файла в килобайтах (вводится с клавиатуры). Вывести размер этого файла в битах.
4. Размер временного интервала задан в минутах. Вывести его величину, выраженную в часах и минутах
5. Размер временного интервала задан в секундах. Вывести его величину, выраженную в часах, минутах и секундах.
6. Дана некоторая сумма денег n в рублях (вводится с клавиатуры). Перевести её в евро по текущему курсу и вывести на экран.
7. Дана некоторая сумма денег n в иенах (вводится с клавиатуры). Перевести её в рубли по текущему курсу и вывести на экран.
8. Дана длина стола N в ярдах (вводится с клавиатуры). Перевести её в метры и вывести на экран.
9. Дан размер диагонали в дюймах (вводится с клавиатуры). Перевести её в сантиметры и вывести на экран.
10. Пусть заданы длины оснований трапеции и ее высота. Найти площадь трапеции.
11. Дана температура воздуха в Фаренгейтах. Перевести её в градусы Цельсия и вывести на экран.
12. Дана температура воздуха в градусах Цельсия. Перевести её в Кельвины и вывести на экран.
13. Даны значения коэффициентов квадратного уравнения a, b и c. Найти его корни
14. Дано расстояние между городами в милях (вводится с клавиатуры). Перевести его в километры и вывести на экран.
15. Даны координаты точки (x,y). Найти расстояние от неё до начала координат.

Задание 3

Пример реализации

```
/*2. Дано трехзначное число. Найти и вывести на печать
произведение цифр заданного числа.*/

//Объявление двух целочисленных значений. "x" вводится
пользователем, а "y" является результатом.
short x, y;
int main() {
    //Ввод значения "x" и сопровождающее его сообщение.
```

```
    cout << "Enter a three digit number: ";
    cin >> x;
    /* Данная строка сравнивает количество цифр в "x" с 3 с помощью округления в меньшую сторону десятичного логарифма по "x" и прибавления 1,так как результатом десятичного логарифма является количество разрядов, кроме первого который и прибавляется поверх результата логарифма.*/
    if ((floor(log10(x)) + 1) == 3) {
        /* Формула умножения отдельных чисел трёхзначного числа через деления "x" и округления получившегося результата в меньшую сторону, а также нахождения остатка после деления "x", где нахождение остатка отсекает цифры слева, а деление и округление справа*/
        y = x % 10 * floor(x % 100 / 10) * floor(x / 100);
        //Вывод результата "y" и сопровождающее его сообщение.
        cout << "The multiplication of every digit of this number is " << y << endl;
    }
    else {
        //Сообщение, выводящееся при вводе нетрёхзначного числа.
        cout << "You've entered a non-three digit number!";
    }
    }
```

Варианты

1. Пусть заданы значения переменных x и y. Напишите программу, которая меняла бы значения этих переменных.
2. Пусть заданы значения переменных x, y и z. Напишите программу, которая меняла бы значения этих переменных местами, так чтобы в x оказалось значение переменной y, в y – значение переменной z, а в z –прежнее значение переменной x.
3. Дано двузначное число. Вывести сумму его цифр.
4. Дано двузначное число. Вывести произведение его цифр.
5. Дано двузначное число. Вывести новое число, в котором количество десятков равно количеству единиц исходного числа, а количество единиц – числу десятков.
6. Дано трехзначное число. Найти и вывести на печать сумму цифр заданного числа.
7. Дано трехзначное число. Найти и вывести на печать произведение цифр заданного числа.
8. Дано четырехзначное число. Написать программу вычисления разности между первой и последней цифрой заданного числа.
9. Дано трехзначное число. В нем зачеркнули последнюю цифру и приписали ее в начале. Вывести новое число.
10. Дано трехзначное число. Найти число, полученное при перестановке первой и второй цифр числа.

11. Из трехзначного числа x (вводится с клавиатуры) вычли его последнюю цифру. Когда результат разделили на 10, а к частному слева приписали последнюю цифру числа x, то получилось число n. Найти число n.

12. Дано двузначное число. Вывести сумму квадратов его цифр.

13. В трехзначном числе x (вводится с клавиатуры) зачеркнули первую цифру. Когда оставшееся число умножили на 10, а произведение сложили с первой цифрой числа x, то получилось число n. Найти число n.

14. Дано четырехзначное число. Написать программу вычисления разности между суммой первой и последней цифры заданного числа и второй и третьей цифрой заданного числа.

15. Дано четырехзначное число. Написать программу вычисления разности между произведениями первой и последней цифры заданного числа и второй и третьей цифрой заданного числа.

1.2. Ветвление и выбор

Задание 1

Определить значение кусочной функции для любого заданного *x*. Аргумент и значение функции могут принимать любые значения, все переменные определить как действительные числа.

Пример реализации

```
/*
y=
-3x-10, если x<-2
-x^2, если -2<=x<=2
3, если x>2
*/

/*Объявление значений с плавающей точкой "x" и "y".
"x" вводится пользователем, а "y" является результатом.*/
double x, y;
int main()
{
    //Ввод значения "x" и сопровождающее его сообщение.
    cout<<"Введите значение X:";
    cin>>x;
    //Начало ветвления.
    //Первое условие.
    if(x<-2){
        //Следующая за выполненным условием операция.
        y=-3*x-10;
    }
    //Второе условие.
    else if(x>=-2 && x<=2){
        //Следующая за выполненным условием операция.
```

```
    y=pow(-x,2);
}
//Третье условие.
else if(x>2){
    //Следующая за выполненным условием операция.
    y=3;
}
//Конец ветвления.
//Вывод результата "y" и сопровождающее его сообщение
cout<<"Результат:"<<y;

return 0;
}
```

Варианты

1. $$y=\begin{cases} x-5,\ x\le 0,\\ \dfrac{\ln x}{x^2+x},\ 0<x<3,\\ \sqrt{x+5},\ x\ge 3. \end{cases}$$

2. $$y=\begin{cases} 8-(x+6)^2, x<-6,\\ |x^2-6|x|+8|, -6\le x\le 5,\\ 3, x>5 \end{cases}$$

3. $$y=\begin{cases} x^2, x<-2,\\ \frac{1}{x^2}, -2\le x\le 0,\\ \sqrt{x}, x>0 \end{cases}$$

4. $$y=\begin{cases} 2x-2, x<3,\\ -3x+13, 3\le x\le 4,\\ 1.5x-7, x>4 \end{cases}$$

5. $$y=\begin{cases} 1, -4\le x\le -2,\\ -0.5x^2+3, -2\le x\le 2,\\ \frac{x}{2}, 2<x\le 4 \end{cases}$$

6. $$y=\begin{cases} x+5, x\le 1,\\ |2(x-3)^2-2|, 1<x\le 4,\\ -\sqrt{x-3}, 4<x\le 7 \end{cases}$$

7. $$y=\begin{cases} -5x-15, x<-2,\\ -x^2, -2\le x\le 2,\\ 10, x>2 \end{cases}$$

8. $$f(x)=\begin{cases} x+2, если -4\le x\le -1\\ x^2, если -1<x\le 0\\ 4, если 0<x\le 4 \end{cases}$$

9. $y = \begin{cases} -x, x < 0, \\ 10, x = 0, \\ x, x > 0 \end{cases}$

10. $y = \begin{cases} x + 2, x < -1, \\ x^2, -1 \leq x \leq 2, \\ 5 - x, x > 2 \end{cases}$

11. $y = \begin{cases} x - 10, x < -3, \\ x^3, -3 \leq x \leq 6, \\ -x, x > 6 \end{cases}$

12. $y = \begin{cases} 2x + 1, \text{ если } x < 0, \\ -1,5x + 1, \text{ если } 0 \leq x < 2, \\ x - 4, \text{ если } x \geq 2 \end{cases}$

13. $y = \begin{cases} x - 0,5, \text{ если } x < -2, \\ -2x - 6,5, \text{ если } -2 \leq x \leq -1, \\ x - 3,5, \text{ если } x > -1 \end{cases}$

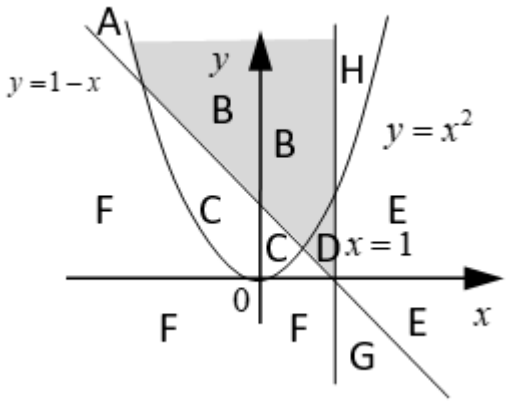
14. $y = \begin{cases} 2, -3 \leq x \leq 1, \\ \sqrt{x}, 1 < x \leq 4, \\ (x + 5)^2 + 1, 4 < x \leq 6 \end{cases}$

15. $y = \begin{cases} 0, x \leq 0, \\ 1/x^2, 0 < x \leq 10, \\ 1, x > 10 \end{cases}$

Задание 2

Написать программу, которая определяет по введенным координатам, попадает ли заданная точка в заштрихованную область. Координаты точки вводятся с клавиатуры. Программа выводит строку «ДА», если точка попадает, и «НЕТ», если точка находится за пределами области/

Пример реализации

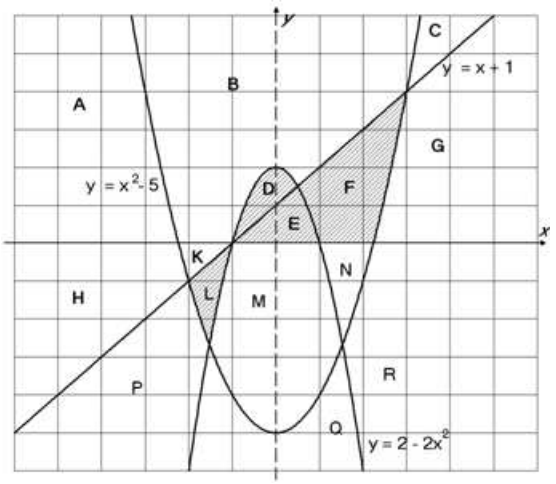



```
/*Объявление значений с плавающей точкой "x" и "y".
Как "x", так и "y" вводятся пользователем.*/
double x, y;
```

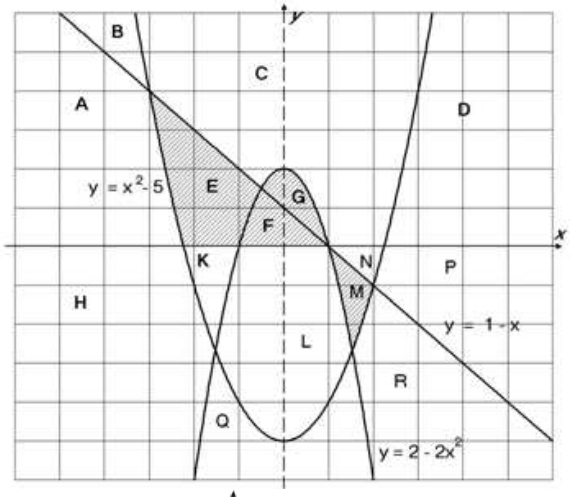
```
int main()
{
	//Ввод значения "x" и сопровождающее его сообщение.
	cout << "Введите значение X:";
	cin >> x;
	//Ввод значения "y" и сопровождающее его сообщение.
	cout << "Введите значение Y:";
	cin >> y;
	//Начало ветвления.
	/*Первое условие(-1.618 ближайшее вещественное значение "x"
к тому,на котором пересекаются два графика).*/
	if (-1.618f <= x <= 1) {
		//Следующее за выполненным условием ветвление.
		if (y >= 1 - x) {
			//Результат.
			cout << "ДА";
		}
		else {
			//Результат.
			cout << "НЕТ";
		}
	}
	/*Второе условие.*/
	else if (x < -1.618f) {
		//Следующее за выполненным условием ветвление.
		if (y >= pow(x, 2)) {
			//Результат.
			cout << "ДА";
		}
		else {
			//Результат.
			cout << "НЕТ";
		}
	}
	else if (x > 1) {
		//Результат.
		cout << "НЕТ";
	}
	//Конец ветвления.

	return 0;
    }
```

Варианты

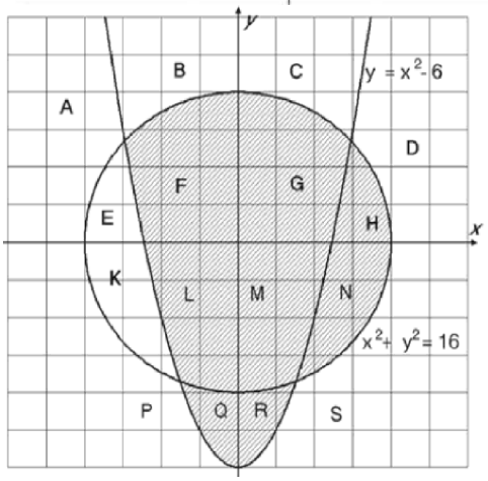
1.



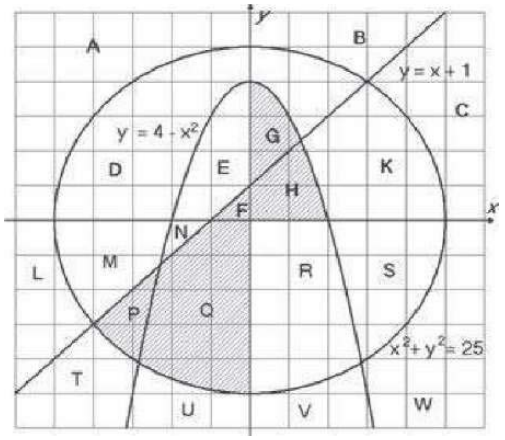
2.



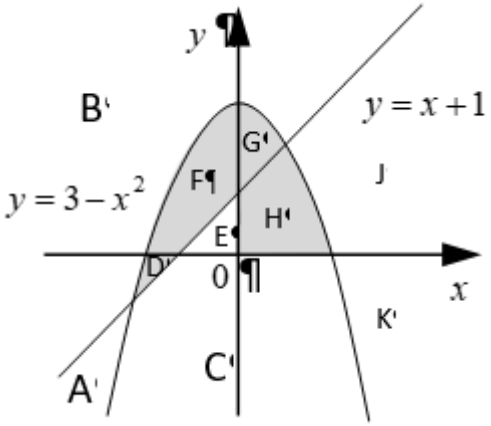
3.



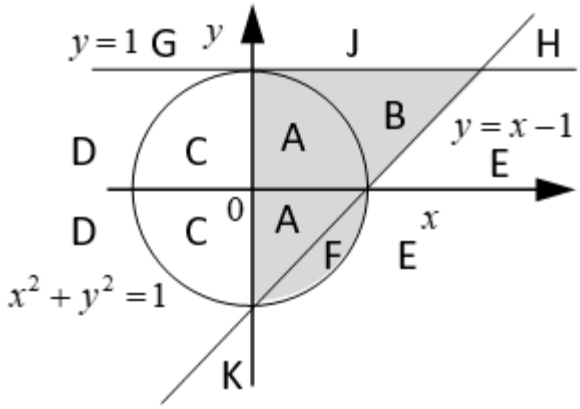
4.


y
x
A
B
C
D
E
F
G
H
K
L
M
N
P
Q
R
S
T
U
V
W
y = x + 1
y = 4 - x²
x² + y² = 25

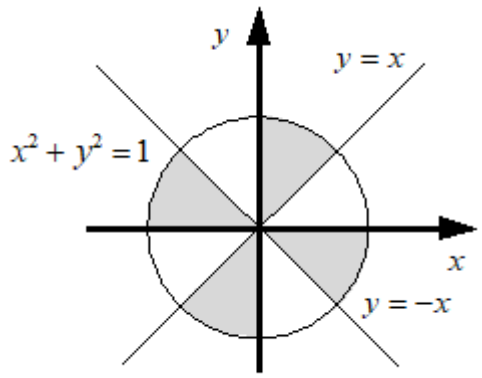
5.


y
x
B'
G'
F
J
H'
E'
D'
0
K'
C'
A'
y = x + 1
y = 3 − x²

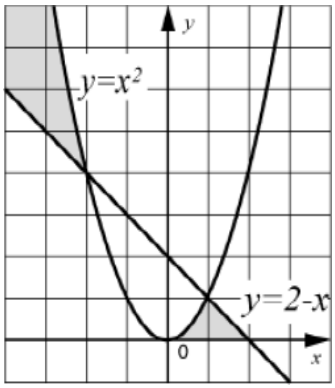
6.


y = 1
G
y
J
H
B
y = x − 1
D
C
A
E
0
x
F
K
x² + y² = 1

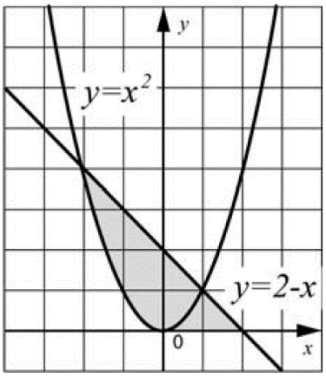
7.


y
y = x
x² + y² = 1
x
y = −x

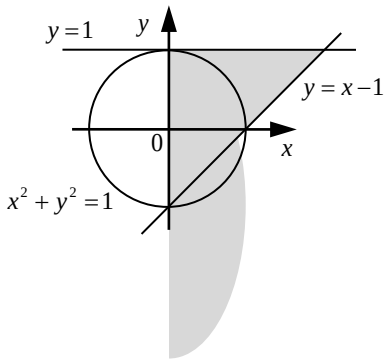
8.


y
y=x²
y=2-x
0
x

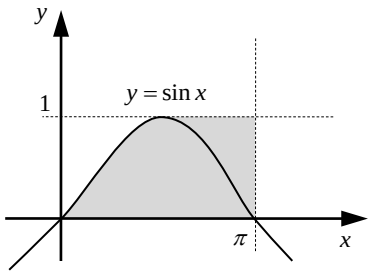
9.


y
y=x²
y=2-x
0
x

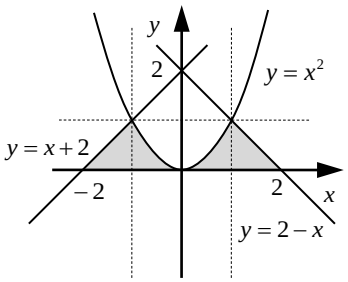
10.


y = 1
y
y = x − 1
0
x
x² + y² = 1

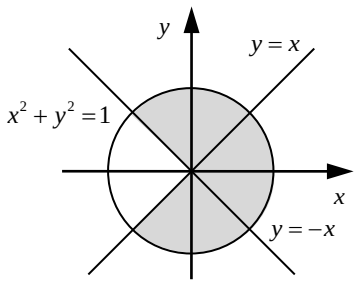
11.


y
1
y = sin x
π
x

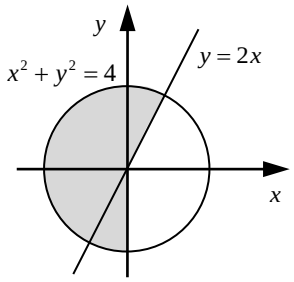
12.


y
2
y = x²
y = x + 2
−2
2
x
y = 2 − x

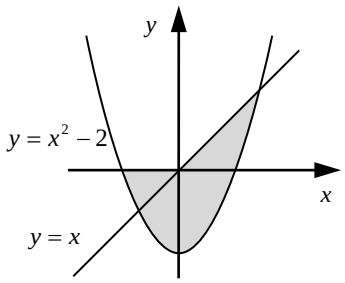
13.



14.



15.



Задание 3

Решить задачи по вариантам с помощью оператора switch().

Пример реализации

```
/*Дано целое число в диапазоне от 1 до 5. Вывести строку - словесное описание соответствующей оценки (1 - «плохо», 2 - «неудовлетворительно», 3 - «удовлетворительно», 4 - «хорошо», 5 - «отлично»)*/
#include<iostream>
#include<stdio.h>
#include<math.h>
using namespace std;
int main()
{
    int p;
    cin>>p;
    switch(p)
    {
  case 1:
    std::cout<<"плохо";
    break;
  case 2:
    std::cout<<"неудовлетворительно";
    break;
  case 3:
    std::cout<<"удовлетворительно";
    break;
```

```
  case 4:
    std::cout<<"хорошо";
    break;
  case 5:
    std::cout<<"отлично";
    break;
    default:
     cout << "неправильный ввод" << endl;
}
    return 0;
}
```

Варианты

1. Дан номер месяца (1 – январь, 2 – февраль, ...). Вывести название соответствующего времени года («зима», «весна» и т. д.).

2. Дано целое число в диапазоне от 0 до 9. Вывести строку – название соответствующей цифры на русском языке (0 – «ноль», 1 – «один», 2 – «два», ...)

3. Арифметические действия над числами пронумерованы следующим образом: 1 – сложение, 2 – вычитание, 3 – умножение, 4 – деление. Дан номер действия и два числа A и B (B не равно нулю). Выполнить над числами указанное действие и вывести результат

4. Единицы длины пронумерованы следующим образом: 1 – дециметр, 2 – километр, 3 – метр, 4 – миллиметр, 5 – сантиметр. Дан номер единицы длины и длина отрезка L в этих единицах (вещественное число). Вывести длину данного отрезка в метрах

5. Составить программу, которая по возрасту человека (вводится с клавиатуры как целое число) определяет его принадлежность к возрастной группе: от 0 до 13 – мальчик; от 14 до 20 – юноша; от 21 до 70 – мужчина; более 70 – старец

6. Локатор ориентирован на одну из сторон света («С» – север, «З» – запад, «Ю» – юг, «В» – восток) и может принимать одну из трех цифровых команд:-1 – поворот налево, 1 – поворот направо, 2 – поворот на 180 градусов. Дан символ C – исходная ориентация локатора и число N – посланная ему команда. Вывести ориентацию локатора после выполнения команды

7. Дано целое число в диапазоне 20 – 69, определяющее возраст (в годах).Вывести строку – словесное описание указанного возраста, обеспечив правильное согласование числа со словом «год», например: 20 – «двадцать лет», 32 – «тридцать два года», 41 – «сорок один год».

8. Составить программу, которая по номеру дня в месяце печатает день недели. Считаем, что 1–е число месяца – понедельник.

9. Составить программу, которая по порядковому номеру месяца определяет, к какому времени года он принадлежит.

10. Составить программу, которая по порядковому номеру месяца определяет количество дней в нем (для не високосного года).

11. Мастям игральных карт присвоены порядковые номера: 1 — пики. 2 — трефы. 3 — бубны. 4 — червы. Достоинству карт, старших десятки, присвоены номера: 11 — валет, 12 — дама, 13 — король, 14 — туз. Даны два целых числа: N — достоинство ($6 < N < 14$) и M — масть карты ($1 < M < 4$). Вывести название соответствующей карты вида «шестерка бубен», «дама червей», «туз треф» и т. п.

12. Дано целое число в диапазоне 10-40, определяющее количество учебных заданий по некоторой теме. Вывести строку-описание указанного количества заданий, обеспечив правильное согласование числа со словами «учебное задание», например: 18 — «восемнадцать учебных заданий», 23 — «двадцать три учебных задания», 31 — «тридцать одно учебное задание».

13. Даны два целых числа: B (день) и M (месяц), определяющие правильную дату. Вывести знак Зодиака, соответствующий этой дате: «Водолей» (20.1-18.2), «Рыбы» (19.2-20.3), «Овен» (21.3-19.4), «Телец» (20.4-20.5), «Близнецы» (21.5-21.6), «Рак» (22.6-22.7), «Лев» (23.7-22.8), «Дева» (23.8-22.9), «Весы» (23.9-22.10), «Скорпион» (23.10-22.11), «Стрелец» (23.11-21.12), «Козерог» (22.12-19.1).

14. В восточном календаре принят 60-летний цикл, состоящий из 12-летних подциклов, обозначаемых названиями цвета: зеленый, красный, желтый, белый и черный. В каждом подцикле годы носят названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. По номеру года определить его название, если 1984 год — начало цикла: «год зеленой крысы».

15. Написать задачу, которая по коду выдаёт основание для выдачи больничного листа (см. Таблицу 1).

Таблица 1

Первый блок (первоначальная причина обращения к врачу)

Код	**Расшифровка**
01	Заболевание
02	Травма
03	Карантин
04	Несчастный случай на производстве или его последствия
05	Отпуск по беременности и родам
06	Протезирование в стационаре
07	Профессиональное заболевание или его обострение
08	Долечивание в санатории
09	Уход за больным членом семьи

Окончание табл. 1

Код	Расшифровка
10	Иное состояние (отравление, проведение манипуляций и др.)
11	Заболевание, указанное в пункте 1 Перечня социально значимых заболеваний (утв. постановлением Правительства РФ от 01.12.04 № 715). Например, туберкулез, гепатиты В, С, сахарный диабет и др.
12	Уход за ребенком в возрасте до 7 лет, если его болезнь включена в специальный перечень (утв. приказом Минздравсоцразвития от 20.02.08 № 84н)
13	Уход за ребенком-инвалидом
14	Уход за ребенком, болезнь которого связана с поствакцинальным осложнением или злокачественным новообразованием (проставляется при согласии сотрудника)
15	Уход за ВИЧ-инфицированным ребенком (проставляется при согласии сотрудника)

1.3. Циклы

Задание 1

Пример реализации

```
/*Для натурального N найти: sin x + sin x^2 + ... + sin x^N,
где x – любое число.*/

int main()
{
    long double b('0'),x('0');
    int n('0');
    //Ввод значения с плавающей запятой "X" и пояснение.
    cout << "Введите X:";
    cin>>x;
    //Обнуление ввода..?
    cin.ignore();
    //Ввод целочисленного значения "N" и пояснение.
    cout << "Введите N:";
    cin>>n;
    //Доводим значение до n+1
    n++;
    //Начало цикла.
for(n>0;--n;){
    //Сумма.
    b+=sin(pow(x,n));
}
//Вывод результата.
cout<<"Результат: "<<b;
    return 0;
}
```

Варианты

1. Дано натуральное число N. Вычислить сумму ряда

$$\left(1+\frac{1}{1^2}\right)\left(1+\frac{1}{2^2}\right)\ldots\left(1+\frac{1}{\mathrm{N}^2}\right)$$

2. Дано натуральное число N. Вычислить сумму ряда:
$$\frac{1}{2}+\frac{3}{4}+\frac{5}{6}+\cdots+\frac{2\mathrm{n}-1}{2\mathrm{n}}$$
3. Даны действительное число a, натуральное число N. Вычислить:
a(a+1)(a+2)(a+3)...(a+N-1).

4. Даны действительное число a, натуральное число N. Вычислить сумму ряда:
$$\frac{1}{\mathrm{a}}+\frac{1}{\mathrm{a(a+1)}}+\cdots+\frac{1}{\mathrm{a(a+1)\ldots(a+n)}}.$$

5. Даны действительное число a, натуральное число N. Вычислить сумму ряда:
$$\frac{1}{\mathrm{a}}+\frac{1}{\mathrm{a}^2}+\frac{1}{\mathrm{a}^4}+\cdots+\frac{1}{\mathrm{a}^{2\mathrm{n}}}.$$

6. Для натурального N найти: $sin\,x+2\,sin^{2!}\,x+\cdots+N\,sin^{N!}\,x$, где x – любое число.
7. Для натурального N найти: $\sin x+\sin x^{2!}+\ldots+\sin x^{N!}$, где x – любое число.
8. Для натурального N найти сумму ряда : $1+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{\mathrm{N}!}$.
9. Для натурального N найти: $\sin\frac{1}{\mathrm{N}}+\sin\frac{1}{2\mathrm{N}}+\cdots+\sin\frac{1}{\mathrm{N}^2}$.
10. Составить программу для вычисления суммы ряда:
$$\sum_{n=1}^{10}\frac{\sqrt[4]{n}}{n+1}$$
11. Составить программу для вычисления суммы ряда:
$$\sum_{n=1}^{15}n+e^{-2n}$$
12. Составить программу для вычисления суммы ряда:
$$\sum_{n=1}^{20}\frac{e^{-n}}{(2\cdot n-1)}$$
13. Составить программу для вычисления суммы ряда:
$$\sum_{n=1}^{40}\frac{1}{(2\cdot n-1)\cdot 2^{2\cdot n-1}}$$

14. Составить программу для вычисления суммы ряда:

$$\sum_{n=1}^{10} \frac{1}{\sqrt[3]{n}\cdot(n+1)\cdot(n+2)^3}$$

15. Для натурального N вычислить:

$$\frac{1}{\sin 1} + \frac{1}{\sin 1+\sin 2} + \cdots + \frac{1}{\sin 1+\sin 2+\cdots+\sin N}.$$

Задание 2

Пример реализации

```
/*Запрашивается число n.
Логической переменной t присвоить значение true,
если в последовательности sin(x^k) (k=1, 2, 3, … n)
есть хотя бы одно отрицательное число, иначе false.*/

int main()
{
    long double x('0');
    int n('0'),k('0');
    bool t=false;
    cout << "Введите X:";
    cin>>x;
    //Обнуление ввода..
    cin.ignore();
    cout << "Введите N:";
    cin>>n;
    //Начало цикла.
for(k<=n;++k;){
    //Проверка результата данного выражения на отрицательность.
    if(sin(pow(x,k))<0){
        t=true;
        break;
    }
}
//Ветвление и вывод результата.
if(t){
    cout<<"Результат: "<<"true";
}
else{
    cout<<"Результат: "<<"false";
}
    return 0;
}
```

Варианты

1. Дано действительное число E (E>0). Необходимо вычислить следующую сумму: $\sum_{n=1}^{\infty}\frac{1}{3^n}\cos 3^{n-1}$. Следует учесть только те слагаемые, в которых множитель $\frac{1}{3^n}$ имеет величину не меньшую, чем E.
2. Дана последовательность, состоящая из дробей: $\frac{1}{1},\frac{4}{2},\frac{7}{3},\frac{10}{4}$ ... Какое минимальное количество элементов последовательности нужно сложить, чтобы сумма превысила заданное число S>1?
3. Дана последовательность, состоящая из дробей: $\frac{1}{1},\frac{3}{2},\frac{5}{3},\frac{7}{4}$ ... Какое минимальное количество элементов последовательности нужно сложить, чтобы сумма превысила заданное число S>1?
4. Дана последовательность, состоящая из дробей: $\frac{1}{2},\frac{3}{4},\frac{5}{6},\frac{7}{8}$ ... Какое минимальное количество элементов последовательности нужно сложить, чтобы сумма превысила заданное число S>1?
5. Число π вычисляется по формуле Грегори следующим образом: $\pi = 4 \cdot \left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\cdots\right)$, причем, чем больше слагаемых в скобках, тем выше точность вычисления числа π. Определить минимальное количество слагаемых для вычисления π с точностью 0.001.
6. Числа Фибоначчи определяются по формулам: $f_0 = 0,\ f_1 = 1,\ f_k = f_{k-2} + f_{k-1}$. Найти первое число Фибоначчи, большее m (m >0).
7. Запрашивается число n. Логической переменной t присвоить значение true, если в последовательности $cos(x^k)$ (k=1, 2, 3, … n) есть хотя бы одно отрицательное число, иначе false.
8. Числа Фибоначчи определяются по формулам: $f_0 = 0,\ f_1 = 1,\ f_k = f_{k-2} + f_{k-1}$. Вычислить сумму всех чисел Фибоначчи, которые не превосходят M (M>0).
9. Найти первый отрицательный элемент последовательности $\sin(ctg(x_i))$, если x_1 – запрашивается, а $x_{i+1} = x_i + 0.3$.
10. Дана последовательность целых чисел (не менее 100). Элементы последовательности генерируются случайным образом в диапазоне [-3; 3]. Вывести первый минимум и его номер (в предположении, что в последовательности несколько минимальных элементов). Задача решается без применения массива.
11. Дана последовательность целых чисел (не менее 100). Элементы последовательности генерируются случайным образом в диапазоне [-30; 70]. Вычислить сумму элементов последовательности, предшествующих

«особому» элементу, то есть элементу, который больше своего соседа справа. Задача решается без применения массива.

12. Дана последовательность, состоящая из дробей: $\frac{1}{1}, \frac{4}{2}, \frac{9}{4}, \frac{16}{8}$ ... Какое минимальное количество элементов последовательности нужно сложить, чтобы сумма превысила заданное число S>1?
13. Составить программу для вычисления количества положительных значений функции $y = \sum_{i=1}^{10} \cos(5i^2 + 8x)$. При x, изменяющемся от 13 до 33 с шагом 1.5, используя цикл do while.
14. Составить программу вычисления функции *chx* по приближенной формуле $chx = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots + \frac{x^{2n+1}}{(2n+1)!}$ с точностью $E \geq \left| \frac{x^{2n+1}}{(2n+1)!} \right|$.
15. Задано положительное число e ($e<10^{-2}$). Последовательность $a_1, a_2, \ldots, a_n$ образуется по следующему закону:

$$a_i = \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{3}\right) \cdot \ldots \cdot \left(1 - \frac{1}{i+1}\right).$$

Найти первый элемент последовательности, для которого выполнено условие $|a_n - a_{n-1}| < e$. Элементы последовательности вывести в 1 столбец. Задачу решить без применения массива.

Задание 3

Пример реализации

```
/* Дано произвольное целое положительное число K (K<≤10^9). Вывести
цифры этого числа в порядке невозрастания (например, 546085 =>
865540). Функции работы со строками не использовать.
*/
#include <iostream>
using namespace std;

int main()
{
	int a, b, min,c ;
	a = rand() ;
cout<<"число"<<a<<endl;
min=10;
	while(a!=0)
	{
		b = a%10;
		a=a/10;
		c=a%10;
		if (b<min)
		{
		    min=b;
		    cout<<”в порядке невозрастания:”<<min;
		}
```

```
            else
            {
                if (c>b)
                cout<<c;
            }
            a = a/10;
        }
        return 0;
}
```

Варианты

1. Даны два натуральных числа m и n ($m \le 9999, n \le 9999$). Проверить, есть ли в записи числа m цифры, совпадающие с цифрами в записи числа n.
2. Дано произвольное целое положительное число K ($K \le 10^9$). Вывести цифры этого числа в порядке неубывания (например, 546085 => 045568). Функции работы со строками не использовать.
3. Дано произвольное целое положительное число K ($K \le 10^9$). Вывести четные цифры этого числа в порядке невозрастания (например, 546085 => 865540). Функции работы со строками не использовать.
4. Дано целое число в диапазоне от 0 до 1000000. Проверить присутствуют ли одновременно в записи числа цифры 1 и 5.
5. Дано произвольное целое положительное число K ($K \le 10^9$). Вывести цифры этого числа в обратном порядке (например, 5485 => 5845). Функции работы со строками не использовать.
6. Дано произвольное целое положительное число K ($K \le 10^9$). Вывести новое число, полученное из K вычеркиванием всех четных цифр (например, 234583 => 353). Функции работы со строками не использовать.
7. Дано произвольное целое положительное число K ($K \le 10^9$). Найти сумму всех четных цифр этого числа. Функции работы со строками не использовать.
8. Дано произвольное целое положительное число K ($K \le 10^9$). Найти сумму всех нечетных цифр этого числа. Функции работы со строками не использовать.
9. Дано произвольное целое положительное число K ($K \le 10^9$). Вывести новое число, полученное из K путем замены последней цифры на значение наибольшей цифры (например, 2854353 => 2354358). Функции работы со строками не использовать.
10. Дано произвольное целое положительное число K ($K \le 10^9$). Найти количество цифр в цифровой записи данного числа, которые имеют наименьшее значение (например, 956569 – количество цифр с наименьшим значением равно 2 – две цифры 5).

11. Дано произвольное целое положительное число K ($K \le 10^9$). Найти сумму всех цифр этого числа, больших заданного T ($0 \le T \le 9$). Функции работы со строками не использовать.
12. Дано целое число в диапазоне от 0 до 1000000. Проверить отсутствуют ли в записи числа четные цифры.
13. Найти на отрезке [n, m] минимальное натуральное число, имеющее наибольшее количество делителей.
14. Дано произвольное целое положительное число K ($K \le 10^9$). Вывести новое число, полученное из K вычеркиванием всех единиц (например, 2134513 => 23453). Функции работы со строками не использовать.
15. Дано натуральное число n. Проверить, есть ли в записи числа три одинаковые цифры ($n \le 9999$).

Задание 4. Повышенный уровень сложности

Пример реализации

```
/*Дана лента билетов, содержащих 6-разрядные номера от N до M
(вводятся с клавиатуры). Определить количество «счастливых»
билетов. («Счастливым» билетом считать билет, сумма первых 3-х
цифр которого равна сумме 3-х последних цифр.)
*/
}#include <iostream>
using namespace std;
int main(){
    int N,M;
    int t=0;
    cout << "Задайте промежуток от N до M: ";
    cin >> N >> M;
    if ((N,M>=100000) and (M,N<1000000) and (N<=M)){
        int k=0,x;
        for (N;N<=M;N++) {
            x=N; t++;
            if
((x/100000)+((x/10000)%10)+((x/1000)%10)==(x%10)+((x%100)/10)+((x%
1000)/100)) k=k+1;
        }
        cout<<"Количество счастливых билетов: "<<k <<"//"<<t;;
    }
    else cout << "Неверно задан числовой промежуток";
}
```

Варианты

1. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить минимальное количество «несчастливых» билетов, расположенных между 2-я «счастливыми»

билетами. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)

2. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить максимальное количество «несчастливых» билетов, расположенных между 2-я «счастливыми» билетами. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)
3. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить количество билетов, «счастливых» и нет, расположенных между первым и последним «счастливыми» билетами в диапазоне. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)
4. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить количество «счастливых» билетов, у которых первая тройка цифр образуют чётное число, а вторая – нечётное. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)
5. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить количество «счастливых» билетов, у которых первая тройка цифр образуют нечётное число, а вторая – чётное. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)
6. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить количество «счастливых» билетов, у которых обе тройки цифр составляют чётные числа. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)
7. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить количество «счастливых» билетов, у которых в номере есть хотя бы одна цифра 2. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)
8. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить количество «счастливых» билетов, номер которых заканчивается на 0. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)
9. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить количество «счастливых» билетов, у

которых в номере первая тройка чисел равна второй, например: 123123. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)

10. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить количество «счастливых» билетов, у которых в номере есть две цифры 5, расположенные в любом порядке («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)
11. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить минимальное количество «несчастливых» билетов, расположенных между 2-я билетами с номерами – палиндромами. (Палиндромом является симметричное число, например: 234432 или 254452
12. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить максимальное количество «несчастливых» билетов, расположенных между 2-я «счастливыми» билетами с номерами – палиндромами. (Палиндромом является симметричное число, например: 234432 или 254452.)
13. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить количество «счастливых» билетов, сумма цифр первой тройки которых не превышает 10. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)
14. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить количество «счастливых» билетов, сумма цифр первой тройки которых больше 10. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)
15. Дана лента билетов, содержащих 6-разрядные номера от N до M (вводятся с клавиатуры). Определить количество «счастливых» билетов, у которых в номере есть хотя бы одна цифра 9. («Счастливым» билетом считать билет, сумма первых 3-х цифр которого равна сумме 3-х последних цифр.)

1.4. Одномерные массивы

Задание 1

Элементы массива генерируются случайным образом в диапазоне [-100; 100].

Пример реализации

```
/*Даны натуральное число n, последовательность натуральных чисел
a0, ..., an в диапазоне от -11 до 45.
Вычислить обратную величину произведения тех элементов ai
последовательности a0,
..., an, для которых выполнено i+1<ai<i!.
*/

#include <iostream>
#include <ctime>

using namespace std;

int fact(int x) { //вычисление факториала
	if (x >= 1)
		return x * fact(x - 1);
	else
		return 1;
}

int main() {
	srand(time(NULL));
	int n=0,proiz=1;
	cout << "введите кол-во чисел: ";
	cin >> n;
	int* arr = new int[n+1]; //т.к. от a0 до an
	cout << "\nПоследовательность чисел: ";
	for (int k = 0; k < n+1; k++) {
		*(arr + k) = rand() % 57 - 11;
		cout << *(arr + k) << " ";
		if (*(arr + k) > k + 1 && *(arr + k) < fact(k))
			proiz *= *(arr + k);
	}
	cout << "\nОбратная величина произведения равна: " << 1 /
proiz;
	return 0;
}
```

Варианты

1. Дан массив целых чисел. Найти сумму 7 наибольших элементов.
2. Заданы действительное число X и массив A[n]. Определить элемент, который по значению ближайший к числу X. Напечатать элемент и его номер.
3. Дан массив чисел (не менее 100 элементов). Найти сумму 5 наименьших элементов.
4. Заданы N целых чисел. Определить, сколько среди них четных и притом делящихся на 7 без остатка. Вывести эти числа.
5. Дана последовательность целых чисел a_0, ..., a_{50}. Получить сумму тех чисел данной последовательности, которые кратны 5, и при этом нечетны и отрицательны.

6. Даны натуральные числа n, p и последовательность целых чисел a_0, ..., a_n. Получить произведение элементов данной последовательности, кратных p.
7. Дана последовательность целых чисел a_0, ..., a_{50}. Получить сумму тех чисел данной последовательности, которые удовлетворяют условию $a_i < i^2$.
8. Даны натуральное число n, последовательность действительных чисел a_0, ..., a_n. Получить удвоенную сумму всех положительных элементов последовательности.
9. Даны натуральное число n, последовательность действительных чисел a_0, ..., a_n. В последовательности a_0, ..., a_n все отрицательные элементы увеличить на 0,5, а все неотрицательные заменить на 0,1.
10. Даны натуральное число n, последовательность действительных чисел a_0, ..., a_n. В последовательности все элементы меньше 2, заменить нулями. Получить сумму элементов, значения которых принадлежит отрезку [3; 7], а также число таких элементов.
11. Даны натуральное число n, последовательность натуральных чисел a_0, ..., a_n. Вычислить сумму тех элементов a_i последовательности a_0, ..., a_n, для которых выполнено $i+1<a_i<i!$.
12. Даны натуральное число n, целые числа a_0, ..., a_n. Получить сумму положительных и число отрицательных четных элементов последовательности.
13. Дан массив целых чисел. Найти, сколько в нем пар одинаковых соседних элементов.
14. Дан массив целых чисел, состоящий из 25 элементов. Найти номер первого отрицательного элемента, делящегося на 5 с остатком.
15. Дана последовательность целых чисел a_0, ..., a_{45}. Получить число отрицательных элементов и число нулевых элементов всей последовательности.

Задание 2

Элементы массива генерируются случайным образом в диапазоне [-100; 100].

Пример реализации

```
/* Элементы массива генерируются случайным образом в диапазоне [-
11; 45].Найти сумму 1-го, 4-го, 9-го, 16-го, …, k-го элементов
одномерного массива X из n элементов (k – наибольшее целое число,
не превышающее  корень из n).
*/

#include <iostream>
#include <cstdlib>
#include <ctime>
#include <math.h>

using namespace std;
```

```
int main() {
	setlocale(0,"");
	srand(time(NULL));
	int n,k=1;
	int sum = 0;
	cout << "Введите число элементов в массиве" << endl;
	cin >> n;
	int* X = new int[n-1];
	for (short i = 0; i < n; i++) {
		X[i] = rand() % 57 - 11;
		cout << X[i]<<endl;
	}
	while (k * k <= sqrt(n)) {
		sum += X[k*k-1];
		k++;
	}
	cout << endl;
	cout << sum;
	return 0;
}
```

Варианты

1. Дан массив действительных чисел A[n]. Получить массив B[n], где B={A0, A_0+A_2, $A_0+A_2+A_3$, $A_0+A_2+A_{3+}$ A_4, … , A_0+A_2, …, A_n}.
2. Дан массив из 25 вещественных чисел. Записать в этот же массив сначала все положительные числа и нули, а затем все отрицательные, сохраняя порядок их следования.
3. Дана последовательность действительных чисел A_0, …, A_{50}. Получить последовательность B={A_2, A_3, …, A_{50}, A_0}.
4. Дан массив A[n]. Получить новый массив, в котором:

$$B(i)=\begin{cases} e^{A(i)}+e^{-A(i)}, & \text{если } A(i)>0; \\ 11, & \text{если } A(i)=0; \\ e^{A(i)}-e^{-A(i)}, & \text{если } A(i)<0. \end{cases}$$

5. Даны натуральное число n, действительные числа A_0, …, A_n. Получить новый массив B[n], в котором

$$B(i)=\begin{cases} A(i), & \text{если } A(i)\ge 0; \\ |A(i)|, & \text{если } A(i)<0. \end{cases}$$

6. Для одномерного массива X из n элементов определить отношение A/B,

где $A=\prod_{k=1}^{n} X_k,\ B=\sum_{k=1}^{n} X_k$.

7. При заданных элементах X_1, X_2, …, X_n и четном n найти значения сумм

$$C=\sum_{i=1}^{\frac{n}{2}} X_i,\ D=\sum_{i=\frac{n}{2}+1}^{n} X_i$$

.

8. Из одномерного массива X из n элементов сформировать массив T по правилу: $T_1 = X_1$; $T_2 = \sqrt{|X_2|}$; $T_3 = \sqrt[3]{|X_3|}$; …; $T_n = \sqrt[n]{|X_n|}$.

9. Дан целочисленный массив с количеством элементов N. Напечатать те элементы, индексы которых являются степенями двойки.
10. При заданных элементах X_1, X_2, …, X_n и четном n найти разность сумм

$$C1 = \sum_{i=1}^{\frac{n}{2}} X_{(2\cdot i-1)},\ C2 = \sum_{i=1}^{\frac{n}{2}} X_{2\cdot i}.$$

11. Найти произведение 1-го, 4-го, 9-го, 16-го, …, k-го элементов одномерного массива X из n элементов (k – наибольшее целое число, не превышающее $\sqrt{n}$).
12. Дан одномерный массив A[n]. Найти: S=max(A_2, …, A_{2k}, …) + min(A_0, A_3, …, A_{2k+1}, …).
13. При заданных коэффициентах A_0, A_2, …, A_n и заданном значении X вычислить значение многочлена $A_0X+A_2X^2+\ldots+A_nX^n$.
14. Найти произведение, сомножители которого представлены каждым третьим элементом: X_1, X_4, X_7, и т. д. – заданного массива X_1, X_2, …, X_n.
15. При заданных координатах C_1, C_2, …, C_n одной и координатах B_1, B_2, …, B_n другой точки n-мерного пространства найти расстояние между ними по формуле:

$$R = \sqrt{(C_1 - B_1)^2 + \ldots + (C_n - B_n)^2}.$$

1.5. Многомерные массивы

Задание 1

Пример реализации

```
/* Дана матрица B. Для каждого столбца с чётным номером вычислить
и напечатать сумму квадратов элементов этого столбца, а для
каждого столбца с нечетным номером вычислить произведение
элементов */
#include<iostream>
#include<ctime>
#include<math.h>
using namespace std;
in tmain()
{
setlocale(LC_ALL,"RUSSIAN");
const int ROW=3;
const int COL=3;
int arr[ROW][COL];
int b=1;
int z=1;
int s=0;
   for (int i=0;i<ROW;i++)
   {
```

```
    for(int j=0;j<COL;j++)
       {
        arr[i][j]=rand() % 20;
        cout<<arr[i][j]<<"\t";
        }
    cout<<endl;
   }
   for (int i=0;i<ROW;i++)
   {
    for(int j=0;j<COL;j++)
       {
         if (j==0) b=b*arr[i][0];
         if (j==1) z=z*arr[i][1];
         if (j==2) s=s+arr[i][2]*arr[i][2];
       }
   }
cout<<b<<endl;
cout<<z<<endl;
cout<<s;

return 0;
}
```

Варианты

1. Задана квадратная матрица размером k×k, составить программу вычисления суммы и произведения элементов матрицы, находящихся на главной диагонали.
2. Составить программу, которая позволяет для матрицы размерностью m×n вычислить произведение положительных, сумму отрицательных и количество нулевых элементов.
3. Дана матрица B. Для каждого столбца с нечётным номером вычислить и напечатать сумму квадратов элементов этого столбца, а для каждого столбца с четным номером вычислить произведение элементов.
4. Найти минимальный элемент j-го столбца матрицы A размером n×n, для которого сумма абсолютных значений элементов максимальна (если таких столбцов несколько взять первый из них).
5. Дан двумерный массив A. Каждый элемент массива, стоящий выше главной диагонали, заменить его квадратом, а ниже диагонали – кубом его значения. Элементы главной диагонали оставить без изменения.
6. Задан двумерный массив B. Найти в нём максимальный элемент и разделить на него каждый элемент массива B.
7. Дана матрица B. Для каждого столбца матрицы вычислить и напечатать разность между квадратом суммы и суммой квадратов элементов этого столбца.
8. Заданы двумерный массив 5×5 и число K. Разделить элементы K-ой строки на диагональный элемент, расположенный в данной строке.
9. Определить и напечатать среднее значение элементов двухмерного массива размером n×m. Найти индексы элемента, наиболее близкого по значению к среднему.

10. Составить программу нахождения минимального положительного элемента матрицы размерностью m×n, а также значение индексов этого элемента.

11. Составить программу для определения количества отрицательных, положительных и равных нулю элементов матрицы R(m×n).

12. 16. Составить программу вычисления суммы и произведения элементов квадратной матрицы A размерностью n×n, расположенных ниже главной диагонали.

13. Составить программу для преобразования квадратной матрицы, состоящего в симметричной относительно главной диагонали перестановки её элементов.

14. Дана действительная матрица размера m×n. Найти значение наибольшего по модулю элемента матрицы, а также индексы первого такого элемента.

15. Дана действительная матрица размера m×n. Найти максимальные элементы по строкам и их сумму.

Задание 2. Повышенный уровень сложности

Данные матрицы задаются случайным образом, если это не оговорено в условии задачи. Промежуточные матрицы выводятся на экран в обязательном порядке. Матрица выводится в виде таблицы, а не строки или столбца.

Пример реализации

```
/*Дано натуральное число n. Получить действительную матрицу {A(i,
j)}, где i и j=1, ..., n, для которой A(i,j)=(1/i)+j ;
•	заменить элементы средних трех столбцов их кубами;
•	вставить между средними столбцами первый столбец;
•	удалить первый столбец, в котором первый элемент больше
последнего;
•	поменять местами средние строки с первой и последней, если n
- четное.

*/

#include <iostream>
#include <math.h>
using namespace std;

int main() {
    int n;
    cout.precision(1);
    cout <<"Введите размер матрицы (n): "; cin>>n;
    double **a=new double *[n+1];
    double **b=new double *[n+1];

    cout<<endl<<"Начальный массив: "<<endl<<endl;
    for (int i = 0; i <= n; i++) {
        a[i]=new double [n+1];
        b[i]=new double [n+1];
```

```
    }
    for (int i = 1; i <= n; i++) {
        for(int j = 1; j <= n; j++)  {
            a[i][j]=(1/double(i))+j;
        }
    }
    for (int i = 1; i <= n; i++) {
        for(int j = 1; j <= n; j++)  {
            cout <<fixed<< (a[i][j]) << "  ";
        }
        cout<<endl;
    }

    cout<<endl<<"1 задание: "<<endl<<endl;
    for (int i = 1; i <= n; i++) {
        for(int j = 1; j <= n; j++)  {
            if((j>=(n/2))and(j<=(n/2)+2)) {
                b[i][j]=pow(a[i][j],3);
                cout << b[i][j] << "  ";
            }
            else cout << a[i][j] << "  ";
        }
        cout<<endl;
    }

    cout<<endl<<"2 задание: "<<endl<<endl;
    for (int i = 1; i <= n; i++) {
        for(int j = 1; j <= n; j++)  {
            if (n%2==0){
                if (j==(n/2)) cout<<a[i][j] << "  "<<a[i][1] << "
";
                else cout<<a[i][j] << "  ";
            }
            else{
                if (j==((n/2)+1)) cout<<a[i][j] << "  "<<a[i][1]
<< "  ";
                else cout<<a[i][j] << "  ";
            }
        }
        cout<<endl;
    }

    cout<<endl<<"3 задание: "<<endl<<endl;
    int j1=0;
    for(int j = 1; j <= n; j++)  {
        if (a[1][j]>a[n][j]) {
            j1=j;
            break;
        }
    }
    if (j1!=0){
        for (int i = 1; i <= n; i++) {
            for(int j = 1; j <= n; j++)  {
```

```
                if (j==j1) continue;
                else cout <<a[i][j]<<"  ";
            }
        cout<<endl;
        }
    }
    else {
        for (int i = 1; i <= n; i++) {
            for(int j = 1; j <= n; j++)  {
                cout << (a[i][j]) << "  ";
            }
            cout<<endl;
        }
    }

    cout<<endl<<"4 задание: "<<endl<<endl;
    if ((n>2)and(n%2==0)){
        for (int i = 1; i <= n; i++) {
            for(int j = 1; j <= n; j++)  {
                if (j==1) b[i][0]=a[i][j];
                if (j==n/2) b[i][1]=a[i][j];
                if (j==(n/2)+1) b[i][2]=a[i][j];
                if (j==n) b[i][3]=a[i][j];
            }
        }
        for (int i = 1; i <= n; i++) {
            for(int j = 1; j <= n; j++)  {
                if (j==1) cout<<b[i][1]<< "  ";
                if (j==n/2) cout<<b[i][0]<< "  ";
                if (j==(n/2)+1) cout<<b[i][3]<< "  ";
                if (j==n) cout<<b[i][2]<< "  ";
                if ((j!=1)and(j!=n/2)and(j!=(n/2)+1)and(j!=n))
cout<<a[i][j]<< "  ";
            }
            cout<<endl;
        }
    }
    else {
        for (int i = 1; i <= n; i++) {
            for(int j = 1; j <= n; j++)  {
                cout << (a[i][j]) << "  ";
            }
            cout<<endl;
        }
    }
    for(int i = 0; i <= n; ++i){
        delete[] a[i];
        delete[] b[i];
    }
    delete[] a;
    delete[] b;
                                    }
```

Варианты

1. Даны целые числа A_0, A_1, A_2. Получить целочисленную матрицу {B(i, j)}, где i и j=0, 1, 2, для которой B[i, j]=Ai.3A_i. Выполнить с матрицей следующие преобразования:

1) заменить все симметричные элементы на 5;
2) вставить перед всеми столбцами, первый элемент которых делится на 5, столбец из нулей;
3) удалить строку, в которой находится первый нечетный отрицательный элемент;
4) поменять местами средние строки.

2. Даны действительные числа A(1), ..., A(10) и натуральные B(1), ..., B(20). Получить действительную матрицу {C(i, j)}, где i=1,..., 20; j=1, ...,10, для которой $C(i, j) = \frac{A(j)}{1 + B(i)}$:

1) заменить все элементы последних трех столбцов их квадратами;
2) вставить между средними строками строку с максимальным элементом;
3) удалить все строки, в которых первый элемент больше последнего;
4) поменять местами средние столбцы.

3. Получить {A(i, j)}, где i=1, ..., 10; j=1, ..., 12 – целочисленную матрицу, для которой A(i, j)=i+2j. Выполнить с матрицей следующие преобразования:

1) заменить все симметричные элементы на нули;
2) вставить перед всеми строками, первый элемент которых делится на 4 , строку из единиц;
3) удалить столбец, в котором находится первый нечетный положительный элемент;
4) поменять местами столбец с минимальным элементом со вторым и последним.

4. Дано натуральное число n. Получить действительную матрицу {A(i, j)}, где i и j=1, ..., n, для которой $A(i, j) = \frac{1}{i} + j$. Выполнить с матрицей следующие преобразования:

1) заменить элементы средних трех столбцов их кубами;
2) вставить между средними строками первую строку;
3) удалить первый столбец, в котором первый элемент больше последнего;
4) поменять местами средние строки с первой и последней, если n - четное.

5. Дано натуральное число n. Получить действительную матрицу

$$A(i,j)=\begin{cases}\sin(i+j), & при\ i<j;\\ 1, & при\ i=j;\\ \sin\left(i+\dfrac{j}{2i}+3\cdot j\right), & в\ остальных\ случаях.\end{cases}$$

Выполнить с матрицей следующие преобразования:

1) заменить максимальный элемент каждой строки на противоположный;
2) вставить после первого столбца с максимальным элементом столбец из нулей;
3) удалить все столбцы, в которых первый элемент больше заданного числа X;
4) поменять местами средние строки, если n - четное.

6. Дана действительная квадратная матрица {A(i, j)}, где i, j=1, ..., n. Получить квадратную матрицу {B(i, j)}, где i, j=1,...,n, для которой

$$B(i,j)=\begin{cases}A(i,j), & при\ j\ge i;\\ -A(i,j), & при\ j<i.\end{cases}$$

Выполнить с матрицей следующие преобразования:

1) заменить все элементы первых трех строк их квадратами;
2) вставить между средними строками строку с минимальным элементом;
3) удалить все столбцы, в которых первый элемент больше третьего;
4) поменять местами средние строки с первым и последним столбцами, если n - четное.

7. Получить действительную матрицу {A(i,j)}, при i, j=1, ..., 7, первая строка которой задается формулой $A(1,j)=2\cdot j+3$, при j = 1, ..., 7, вторая $A(2,j)=\dfrac{3j}{2}+\dfrac{1}{j}$, при j = 1, ..., 7, а каждая следующая строка есть сумма двух предыдущих. Выполнить с матрицей следующие преобразования:

1) заменить минимальный элемент каждой строки на противоположный;
2) вставить после столбцов с максимальными элементами столбец из нулей;
3) удалить первый столбец, в котором первый элемент больше заданного числа X;
4) поменять местами крайние столбцы.

8. Даны натуральное число m, действительная матрица размера m×6. Выполнить с матрицей следующие преобразования:

1) найти среднее геометрическое:

- каждого из столбцов;

- каждой из строк, имеющих нечетные номера.

2) заменить максимальный элемент каждой строки на противоположный (по знаку);

3) удалить первый столбец, в котором второй элемент меньше заданного числа X.

9. Дано натуральное число n. Выяснить, сколько положительных элементов содержит матрица {A(i, j)}, при i, j=1, ..., n, если $A(i, j) = \sin\left(i + \frac{j}{2}\right)$.

Выполнить с матрицей следующие преобразования:

1) заменить максимальный элемент каждого столбца на его квадрат с противоположным знаком;
2) вставить после столбцов с минимальным элементами столбец из нулей;
3) удалить первую строку с нечетной суммой элементов;
4) поменять местами средние столбцы, если n - четное.

10. Дано натуральное число n. Выяснить, сколько положительных элементов содержит матрица {A(i, j)}, при i, j=1, ..., n, если $A(i, j) = \cos(i^2 + n)$. Выполнить с матрицей следующие преобразования:

1) заменить все элементы столбцов, начинающихся с положительных элементов их квадратами;
2) вставить между средними строками столбец с минимальным элементом;
3) удалить первую строку, в которых первый элемент больше последнего;
4) поменять местами третий и четвертый столбцы с первым и последним.

11. Дана действительная матрица размера n×m, в которой не все элементы равны нулю. Получить новую матрицу путем деления всех элементов данной матрицы на ее наибольший по модулю элемент. Выполнить с матрицей следующие преобразования:

1) найти количество элементов в каждой строке, больших среднего арифметического элементов данной строки;
2) вставить перед столбцом, содержащий минимальный элемент столбец с нулевыми элементами;
3) удалить строку, в которой первый элемент больше последнего;
4) поменять местами третий и четвертый столбцы с первым и последним.

12. Дана действительная квадратная матрица порядка 12. Выполнить с матрицей следующие преобразования:

1) заменить все элементы последних трех столбцов их квадратами;

2) вставить между средними строками строку с максимальным элементом;
3) удалить первую строку, в которых первый элемент больше последнего;
4) начиная с k-го столбца, сдвинуть их вперед, а первые k поставить на место последних, где k≤12.

13. Даны действительные числа x(1), ..., x(8). Получить действительную квадратную матрицу порядка 8:

$$\begin{matrix} x(1) & x(2) & \dots & x(8) \\ x(1)^2 & x(2)^2 & \dots & x(8)^2 \\ \dots & \dots & \dots & \dots \\ x(1)^8 & x(2)^8 & \dots & x(8)^8 \end{matrix}.$$

Выполнить с матрицей следующие преобразования:
1) поменять местами 2 и 4 строку;
2) в каждой строке переставить первый отрицательный и последний положительный, если таких нет, то сообщить об этом;
3) удалить первую строку, в которой первый элемент отрицательный;
4) найти количество четных элементов с четными индексами, подсчитать их сумму.

14. Даны натуральное число n, действительная матрица {A(i, j)}, где i=1, ..., n и j=1, ..., n. Получить последовательность элементов главной диагонали A(1,1), A(2,2), ..., A(n,n)). Выполнить с матрицей следующие преобразования:
- Первую строку поменять местами с последней;
- вставить второй столбец с минимальным элементом;
- удалить первый столбец с минимальной суммой элементов;
- подсчитать сумму и количество отрицательных элементов кратных числу p.

15. Дана действительная матрица размера m×n. Определить числа B(1), ..., B(m), равные соответственно разностям наибольших и наименьших значений элементов строк. Выполнить с матрицей следующие преобразования:
- заменить максимальный элемент каждого столбца на его квадрат с противоположным знаком;
- вставить после столбцов с минимальным элементом столбец из нулей;
- удалить первую строку с нечетной суммой элементов;
- поменять местами второй и предпоследний столбец.

1.6. Строки

Задание 1
Пример реализации

```
/*Дана строка символов. Слова в строке отделяются друг от друга
одним пробелом. Написать программу, вычисляющую среднюю длину слов
в строке. */
#include<iostream>
#include<string>
using namespace std;

int main()
{
    int b=0;
 string text;
cout<<"Введите строку\n";
getline(cin,text,'\n');
int i=0;
int str=text.length();
for (int k=0;k<str;k++)
{
    b++;
if (text[k]==' ')
{
i++;
b--;
}
}
cout<<"Количество слов в тексте - "<<i+1<<endl;
cout<<"Количество символов в тексте без пробелов - "<<b<<endl;
cout<<"Средняя длина слов в тексте - "<<b/(i+1);
return 0;
}
```

Варианты

1. Дана строка символов. Слова в строке отделяются друг от друга одним пробелом. Вывести самое длинное слово.
2. Дана строка символов. Слова в строке отделяются друг от друга одним пробелом. Вывести самое короткое слово.
3. Дана строка символов. Слова в строке отделяются друг от друга одним пробелом. Написать программу, вычисляющую среднюю длину слов в строке.
4. Дана строка символов. Слова в строке отделяются друг от друга одним пробелом. Удалить из строки самое длинное слово.
5. Дана строка символов. Слова в строке отделяются друг от друга одним пробелом. Удалить из строки самое короткое слово.
6. Дана строка символов. Слова в строке отделяются друг от друга одним пробелом. Заменить в строке одно заданное слово (если оно есть) другим.

7. Дана строка символов. Слова в строке отделяются друг от друга любым количеством пробелов. Преобразовать строку таким образом, чтобы слова отделялись строго одним пробелом.

8. Дана строка символов. Слова в строке отделяются одним пробелом. Поменять местами самое длинное и самое короткое слово.

9. Дана строка из нескольких слов. Слова отделяются друг от друга пробелами или запятыми. Подсчитать количество слов, длина которых больше заданного числа.

10. Дана строка из нескольких слов. Слова отделяются друг от друга пробелами или запятыми. Подсчитать количество слов, длина которых меньше заданного числа.

11. Дана строка из нескольких слов. Слова отделяются друг от друга пробелами или запятыми. Вывести все слова, длина которых больше заданного числа.

12. Дана строка из нескольких слов. Слова отделяются друг от друга пробелами или запятыми. Вывести все слова, длина которых меньше заданного числа.

13. Дана строка из нескольких слов. Слова отделяются друг от друга пробелами или запятыми. Подсчитать количество слов, начинающихся и заканчивающихся одной и той же буквой.

14. Дана строка из нескольких слов. Слова отделяются друг от друга пробелами или запятыми. Вывести слова, начинающиеся и заканчивающиеся одной и той же буквой.

15. Дана строка из нескольких слов. Слова отделяются друг от друга пробелами или запятыми. Подсчитать количество слов, начинающихся на гласную букву

Задание 2

Код Цезаря заменяет одну букву другой, отстоящей от нее на заданное количество позиций в алфавите. Например, при сдвиге, равном 1, буква А заменяется на Б, Б — на В, …, Я — на А

Пример реализации

```
/*Дан текст, состоящий из N (2<=N<=10) строк с максимальной длиной 80 символов.
Необходимо вывести этот текст, зашифрованный кодом Цезаря.Размер сдвига символов
принять равным остатку от деления количества слов в предложении на номер буквы в
слове плюс единица.
Например, если в предложении 8 слов, слово ДОМ – шифруется как ЕПР.
Считать, что текст написан синтаксически грамотно,  в качестве знаков
препинания используются точка и запятая,  слова состоят только из букв, перенос
слов по слогам отсутствует.*/
#include <iostream>
#include <string.h>
using namespace std;

int AmountOfWords(char* b)
{
    int slovo, count = 0;
    int i = 0;
    while (b[i] == ' ' && b[i] != '\0')
        i++;
```

```
    slovo = 0;
    while (b[i] != '\0')
    {
        if (b[i] != ' ' && slovo == 0)
        {
            slovo = 1;
            count++;
        }
        else if (b[i] == ' ')
            slovo = 0;
        i++;
    }
    return count;
};

int main()
{
    const int n = 2;
    setlocale(LC_ALL, "Russian");

    if ((n >= 2) and (n < 10))
    {
        char a[n][80];
        for (int i = 0; i < n; i++)
        {
            cout<<"ВВедите строку "<<i+1<<endl;
            cin.getline(a[i], 80, '\n');
        }
        for (int i = 0; i < n; i++)
        {
            int m = 0;
            for (int j = 0; j < strlen(a[i]); j++)
            {
                if ((a[i][j] != ' ') and (a[i][j] != ',') and (a[i][j] != '\0'))
                {
                    m++;
                    int b = int(a[i][j]) + (AmountOfWords(a[i]) % m) + 1;
                    if (((int(a[i][j]) >= 97) and (int(a[i][j]) <= 122)) or
((int(a[i][j]) >= 65) and (int(a[i][j]) <= 90)))
                    {
                        if ((b > 122) or ((b > 90) and (b < 97)))
                        {
                            b = b - 26;
                            a[i][j] = char(b);
                        }
                        else
                        {
                            a[i][j] = char(b);
                        }
                    }
                }
                else
                {
                    m=0;
                }
            }
        }
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < strlen(a[i]); j++)
            {
                cout << a[i][j];
            }
            cout << endl;
```

```
        }
    }
    else
    {
        cout << "Количество строк не входит в диапозон" << endl;
    }
}
```

Варианты

1. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести этот текст, зашифрованный кодом Цезаря. Размер сдвига символов принять равным **номеру буквы в слове**. Например, слово ДОМ шифруется как ЕРП. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.
2. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести этот текст, зашифрованный кодом Цезаря. Размер сдвига символов принять равным **K**. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.
3. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести этот текст, зашифрованный кодом Цезаря. Размер сдвига символов принять равным **номеру буквы в предложении**. Например, если предложение начинается со слова ДОМ, то оно шифруется как ЕРП. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.
4. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести этот текст, зашифрованный кодом Цезаря. Размер сдвига символов принять равным **остатку от деления длины слова на номер буквы в слове плюс единица**. Например, слово ДОМ — шифруется как ЕРН. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.
5. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести этот текст, зашифрованный кодом Цезаря. Размер сдвига символов принять равным **остатку от деления количества слов в предложении на номер буквы в слове**. Например, если в предложении 8 слов, слово ДОМ — шифруется как ЕПР. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.
6. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Составить программу для расшифровки текста, зашифрованного кодом Цезаря, где размер сдвига символов равен **номеру буквы в слове**. Считать, что текст написан синтаксически грамотно, в качестве знаков

препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

7. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Составить программу для расшифровки текста, зашифрованного кодом Цезаря, где размер сдвига символов равен **номеру буквы в предложении**. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

8. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Составить программу для расшифровки текста, зашифрованного кодом Цезаря, где **размер сдвига символов равен К**. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

9. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Составить программу для расшифровки текста, зашифрованного кодом Цезаря, где размер сдвига символов равен **остатку от деления длины слова на номер буквы в слове плюс единица**. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

10. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Составить программу для расшифровки текста, зашифрованного кодом Цезаря, где **размер сдвига символов равен остатку от деления длины слова на номер буквы в слове минус единица**. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

11. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести этот текст, зашифрованный кодом Цезаря. Размер сдвига символов принять равным **1**. Например, слово КОШКА — шифруется как ЛПЩЛБ. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

12. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Составить программу для расшифровки текста, зашифрованного кодом Цезаря, где размер сдвига символов равен **сумме сдвигов двух предыдущих букв**. Для первой буквы принять сдвиг равным 0, а для второй — 1. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

13. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести этот текст, зашифрованный кодом Цезаря. Размер сдвига символов принять равным **номеру символа в алфавите**. Считать, что текст написан синтаксически грамотно, в качестве знаков

препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

14. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Составить программу для расшифровки текста, зашифрованного кодом Цезаря, где размер сдвига символов равен **номеру символа в алфавите**. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

15. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Составить программу для расшифровки текста, зашифрованного кодом Цезаря, где размер сдвига символов равен **сумме сдвигов двух предыдущих букв**. Для первой буквы принять сдвиг равным 1, а для второй — 2. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

Задание 3. Повышенный уровень сложности

Пример реализации

```
/* Дан текст, состоящий из 2 строк с максимальной длиной 80
символов. Необходимо вывести слова, присутствующие в обеих строках
одновременно. Считать, что текст написан синтаксически грамотно, в
качестве знаков препинания используются точка и запятая, слова
состоят только из букв, перенос слов по слогам отсутствует */

#include <string>
#include <iostream>
#include <string.h>

int main()
{
std::string s;
std::string s1;
    char p[1000];
int k, f;
printf("введите первую строку: ");
std::getline(std::cin,s);
printf("введите вторую строку: ");
std::getline(std::cin,s1);
int str=s.length();
int str2=s1.length();
for (k=0;k<str;k++)
{
   for (int f=0;f<str2;f++)
   {
      if (s[k]==s1[f]&&s[k+1]==s1[f+1]&&s[k-1]==s1[f-1])
      {
std::cout<<s[k];
      }
   }
}
```

```
}
```

Варианты

1. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в алфавитном порядке последние слова всех предложений. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.
2. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в обратном алфавитном порядке первые слова всех предложений. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.
3. Дан текст, состоящий из 2 строк с максимальной длиной 80 символов. Необходимо вывести в алфавитном порядке слова, присутствующие в обеих строках одновременно. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует
4. Дан текст, состоящий из 2 строк с максимальной длиной 80 символов. Необходимо вывести в обратном алфавитном порядке те слова, которые не присутствуют в обеих строках одновременно. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует
5. Дан текст, состоящий из 3 строк с максимальной длиной 80 символов. Необходимо вывести все предложения данного текста в порядке неубывания их длины. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует
6. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в обратном алфавитном порядке те слова, где количество гласных превышает количество согласных. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует
7. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в алфавитном порядке те слова, где количество согласных превышает количество гласных. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует
8. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в алфавитном порядке те слова, длина которых превышает K символов. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует
9. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в обратном алфавитном порядке те слова,

длина которых не превышает K символов. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

10. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в обратном алфавитном порядке те слова, последняя буква которых является гласной. Считать, текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

11. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в алфавитном порядке те слова, последняя буква которых является согласной. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует

12. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в алфавитном порядке те слова, первая буква которых является согласной, а последняя гласной. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

13. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в алфавитном порядке предпоследние слова всех предложений. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует, минимальная длина предложений — два слова

14. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в алфавитном порядке вторые слова всех предложений. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует, минимальная длина предложений — два слова.

15. Дан текст, состоящий из N (2≤N≤10) строк с максимальной длиной 80 символов. Необходимо вывести в алфавитном порядке слова, начинающиеся с прописных букв. Считать, что текст написан синтаксически грамотно, в качестве знаков препинания используются точка и запятая, слова состоят только из букв, перенос слов по слогам отсутствует.

1.7. Структуры

ОБЩЕЕ ЗАДАНИЕ

Составить программу, которая обеспечивает ввод из файла «Данные» информации о составе группы, ее просмотр в виде таблицы, а также вывод информации на экран монитора согласно конкретному варианту. В случае если в группе нет студентов с требуемыми данными, выдать соответствующее сообщение.

Информацию о каждом студенте оформить в виде структуры, а совокупность структур объединить в массив.

Пример реализации

```
/* Определить время года, в которое родилось большинство
троечников и вывести список
 *  таких студентов */
#define _CRT_SECURE_NO_WARNINGS

#include <iomanip>
#include <iostream>
#include <fstream>
#include <windows.h>
#include <string>
#include <cmath>
#include <algorithm>
#include <sstream>

using namespace std;

// Имя файла кладем в константу
const char* FILE_NAME = "students.txt";

// Создаем структуру с полями
struct Student
{
    char name[32];
    char surname[32];
    int day;
    int month;
    int year;
    int marks[5];
};

// Считаем студентов в файле
int countLines(string Path)
{
    ifstream file;
    file.open(Path);

    string str;
    int line = 0;

    while (!file.eof())
    {
        getline(file, str);
        line++;
    }

    file.close();

    return line - 1;
}
```

```
int main()
{
    // Устанавливаем кодировку для консоли UTF-8 для корректного
//чтения из файла
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);

    ifstream file;

    string str;
    int line = countLines(FILE_NAME); // Кол-во студентов
    int temp = 0;

    Student* studentArray = new Student[line]; // Создаем массив
структур

    file.open(FILE_NAME); // Открываем файл

    for (int i = 0; i < line; i++)
    {
        getline(file, str); // Берем строку
        replace(str.begin(), str.end(), '-', ' '); // Заменяем
//ненужные символы

        istringstream ss(str); // Преобруем строку в поток
        string word;
        ss >> word; // вытаскиваем первое слово до пробела и т. д.
        strcpy(studentArray[i].surname, word.c_str()); // Копируем
//полученное слово в поле структуры
        ss >> word;
        strcpy(studentArray[i].name, word.c_str());
        ss >> word;
        studentArray[i].year = atoi(word.c_str()); // Преобразуем
//полученное слово в число и кладем в нужное поле
        ss >> word;
        studentArray[i].month = atoi(word.c_str());
        ss >> word;
        studentArray[i].day = atoi(word.c_str());

        for (int j = 0; j < 5; j++)
        {
            ss >> word;
            studentArray[i].marks[j] = atoi(word.c_str());
        }
    }

    // Выводим всех студентов в табличке
    for (int i = 0; i < line; i++)
    {
        cout << "Date: " << setw(2) << studentArray[i].day << ".";
        cout << setw(2) << studentArray[i].month << ".";
        cout << studentArray[i].year << "  ";
        cout << "Marks: ";
```

```
        for (int j = 0; j < 4; j++)
        {
            cout << studentArray[i].marks[j] << ", ";
        }
        cout << studentArray[i].marks[4] << "  ";
        cout << studentArray[i].surname << " " <<
studentArray[i].name;
        cout << endl;
    }

    // Ищем в каком месяце родилось больше всего троечников
//(минимум одна оценка 3)
    cout << endl << "Most students with a grade of '3': ";

    // Создаем массив времен года
    int timeOfYear[4] = { 0, 0, 0, 0 };

    // Создаем флаг для цикла
    bool mark_3 = false;

    for (int i = 0; i < line; i++)
    {
        mark_3 = false;

        // Если есть хоть одна 3, флаг = true
        for (int j = 0; j < 5; j++)
        {
            if (studentArray[i].marks[j] == 3)
            {
                mark_3 = true;
            }
        }
        // Если флаг = true, записываем время года, когда родился
//студент и запоминаем его индекс
        if (mark_3)
        {
            if (studentArray[i].month < 3 || studentArray[i].month
== 12)
            {
                timeOfYear[0] += 1;
            }
            if (studentArray[i].month > 2 && studentArray[i].month
< 6)
            {
                timeOfYear[1] += 1;
            }
            if (studentArray[i].month > 5 && studentArray[i].month
< 9)
            {
                timeOfYear[2] += 1;
            }
            if (studentArray[i].month >= 9 &&
studentArray[i].month <= 11)
```

```
            {
                timeOfYear[3] += 1;
            }
        }
    }

    // Ищем идекс времени года, когда было больше всего троечников
    int max = 0;
    int indexMax = -1;

    for (int i = 0; i < 4; i++)
    {
        if (timeOfYear[i] >= max)
        {
            max = timeOfYear[i];
            indexMax = i;
        }
    }

    // Выводим время года и студентов троечников
    switch (indexMax)
    {
    case 0:
        cout << "Winder" << endl;
        for (int i = 0; i < line; i++)
        {
            mark_3 = false;
            // Если есть хоть одна 3, флаг = true
            for (int j = 0; j < 5; j++)
            {
                if (studentArray[i].marks[j] == 3)
                {
                    mark_3 = true;
                }
            }
            // Если флаг = true, записываем время года, когда
//родился студент и запоминаем его индекс
            if (mark_3)
            {
                if (studentArray[i].month < 3 ||
studentArray[i].month == 12)
                {
                    cout << studentArray[i].surname << " " <<
studentArray[i].name;
                    cout << endl;
                }
            }
        }
        break;
    case 1:
        cout << "Spring" << endl;
        for (int i = 0; i < line; i++)
        {
```

```
            mark_3 = false;
            // Если есть хоть одна 3, флаг = true
            for (int j = 0; j < 5; j++)
            {
                if (studentArray[i].marks[j] == 3)
                {
                    mark_3 = true;
                }
            }
            // Если флаг true, записываем время года, когда
//родился студент и запоминаем его индекс
            if (mark_3)
            {
                if (studentArray[i].month > 2 &&
studentArray[i].month < 6)
                {
                    cout << studentArray[i].surname << " " <<
studentArray[i].name;
                    cout << endl;
                }
            }
        }
        break;
    case 2:
        cout << "Summer" << endl;
        for (int i = 0; i < line; i++)
        {
            mark_3 = false;
            // Если есть хоть одна 3, флаг = true
            for (int j = 0; j < 5; j++)
            {
                if (studentArray[i].marks[j] == 3)
                {
                    mark_3 = true;
                }
            }
            // Если флаг = true, записываем время года, когда
родился студент и запоминаем его индекс
            if (mark_3)
            {
                if (studentArray[i].month > 5 &&
studentArray[i].month < 9)
                {
                    cout << studentArray[i].surname << " " <<
studentArray[i].name;
                    cout << endl;
                }
            }
        }
        break;
    case 3:
        cout << "Autumn" << endl;
        for (int i = 0; i < line; i++)
```

```
        {
            mark_3 = false;
            // Если есть хоть одна 3, флаг = true
            for (int j = 0; j < 5; j++)
            {
                if (studentArray[i].marks[j] == 3)
                {
                    mark_3 = true;
                }
            }
            // Если флаг = true, записываем время года, когда
родился студент и запоминаем его индекс
            if (mark_3)
            {
                if (studentArray[i].month >= 9 &&
studentArray[i].month <= 11)
                {
                    cout << studentArray[i].surname << " " <<
studentArray[i].name;
                    cout << endl;
                }
            }
        }
        break;
    }

    delete[] studentArray;

    file.close();
    return 0;
    }
```

Варианты

1. Вывести анкетные данные студентов-отличников.
2. Вывести анкетные данные всех студентов, у которых по всем предметам «удовлетворительно».
3. Вывести анкетные данные всех студентов, успевающих на «хорошо» и «отлично».
4. Вывести анкетные данные студентов, получивших по одному из предметов оценку «хорошо», а по всем другим – «отлично».
5. Вывести список студентов, фамилии которых начинаются с гласной буквы и их оценки по всем предметам.
6. Вывести список студентов, фамилии которых начинаются с согласной буквы и их даты рождения.
7. Вывести фамилии и даты рождения студентов, не получивших ни одной оценки «удовлетворительно» по всем предметам.
8. Упорядочить список студентов по среднему баллу и вывести весь список.
9. Вычислить средний балл группы и вывести список студентов, имеющих средний балл выше, чем в целом по группе (по всем предметам).

10. Вычислить средний балл группы и вывести список студентов, имеющих средний балл ниже, чем в целом по группе (по всем предметам).

11. Вычислить средний балл группы и вывести список студентов, имеющих средний балл, близкий к среднему баллу группы (отличающийся не более чем на K процентов).

12. Упорядочить список студентов по дате рождения и вывести весь список.

13. Вывести по отдельности средний балл за сессию юношей и девушек

14. Определить время года, в которое родилось большинство успевающих студентов (тех, кто учится без троек) и ввести список таких студентов

15. Определить время года, в которое родилось большинство двоечников и вывести список таких студентов

1.8. Функции

Задание 1.

Пример реализации

```
/*
Дан одномерный массив из 50 случайных целых чисел в диапазоне от
10 до 85 включительно.
Вывести в порядке возрастания те числа из данного диапазона,
которые ни разу не встречаются в массиве.
Создать функцию для поиска элемента в массиве.
*/
#include <iostream>
#include <windows.h>
#include <ctime>

using namespace std;

int f(int *A) { //подсчёт кол-ва встречающихся букв
	int i = 10;
	int m=0;
	while (i <= 85) {
		for (int k = 0; k < 50; k++) {
			if (i == A[k])
				break;
			if (k == 49)
				m++;
		}
		i++;
	}
	return m;
}

void g(int* A,int *S,int N) { //заполнение массива числами,
которых нет
	int i = 10;
	int m = 0;
```

```
	while (i <= 85) {
		for (int k = 0; k < 50; k++) {
			if (i == A[k])
				break;
			if (k == 49) {
				S[m] = i;
				m++;
			}
		}
		i++;
	}
}

void sort(int* x, int len) { //соортировка массива
	//сортировка массива по возрастанию
	int t;
	len--;
	while (len > 1) {
		for (int j = 0; j < len; j++) {
			if (x[j] > x[j + 1]) {
				t = x[j];
				x[j] = x[j + 1];
				x[j + 1] = t;
			}
		}
		len--;
	}
}

int main() {
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);
	srand(time(NULL));
	int* arr = new int[50]; //массив чисел
	cout << "первоначальный массив."<<endl;
	for (int i = 0; i < 50; arr[i] = rand() % 76+10,cout <<
arr[i]<<" ", i++); //заполнение массива случайными числами
	cout << endl;
	int number = f(arr);
	int* s = new int[number]; //новый массив для чисел, которых
нет в массиве arr
	g(arr,s,number);
	sort(s, number);
	cout << "------------------------------" << endl;
	cout << "Массив чисел, которых нет." << endl;
	for (int k = 0; k < number; k++)
		cout << s[k] << " ";
	return 0;
    }
```

Варианты

1. Дано *N* десятков целых чисел. Определить, сколько из них могут составлять геометрическую прогрессию. Проверку оформить в виде функции.

2. Дано N десятков целых чисел. Определить, сколько из них могут составлять арифметическую прогрессию. Проверку оформить в виде функции.

3. Дано N десятков целых чисел. Определить, сколько из них могут составлять ряд Фибоначчи. Первое число Фибоначчи равно 0, второе – 1. Каждое последующее равно сумме двух предыдущих. Проверку оформить в виде функции.

4. Дано N пар чисел, представляющих собой координаты точек на плоскости. Найти R – радиус наименьшей окружности с центром в начале координат, в которую попадают все точки. Определение расстояния от точки до начала координат оформить в виде функции.

5. Известны оценки группы студентов за сессию. В группе 20 студентов, в сессии 4 экзамена. Определить суммарную стипендию. Считать, что стипендия в размере R рублей начисляется студентам, сдавшим сессию без троек, а отличники получают стипендию, повышенную на 25% . Подсчет стипендии студента оформить в виде функции.

6. Известен расход электроэнергии по всем квартирам 24-х квартирного дома. Определить суммарную плату за электричество. При расходе до 100 кВт*ч на человека берется тариф R рублей за 1 кВт*ч, в случае превышения нормы тариф возрастает на 20%. Подсчет платы для квартиры оформить в виде функции.

7. Известна ежемесячная заработная плата персонала предприятия в течение календарного года. Вывести фамилии тех сотрудников, у которых годовая заработная плата выше средней. Считать, что штат предприятия составляет 7 человек. Подсчет годовой зарплаты работника оформить в виде функции.

8. Известна ежемесячная заработная плата персонала предприятия в течение календарного года. Вывести фамилии сотрудников с минимальной и максимальной годовой заработной платой. Считать, что штат предприятия составляет 8 человек. Подсчет годовой зарплаты работника оформить в виде функции.

9. Дан одномерный массив из 100 случайных целых чисел в диапазоне от 5 до 25 включительно. Вывести все числа, которые максимально часто встречаются в массиве и количество их повторений. Подсчет количества повторений для числа оформить в виде функции.

10. Дан одномерный массив из 150 случайных целых чисел в диапазоне от 14 до 37 включительно. Вывести те числа, которые наиболее редко встречаются в массиве и количество их повторений. Подсчет количества повторений для числа оформить в виде функции.

11. Дан одномерный массив из 100 случайных целых чисел в диапазоне от 0 до 100 включительно. Вывести в порядке возрастания те числа из данного диапазона, которые встречаются в массиве ровно 2 раза. Создать функцию для поиска элемента в массиве.

12. Дан одномерный массив из 40 случайных целых чисел в диапазоне от 16 до 89 включительно. Вывести минимальное и максимальное числа из

данного диапазона, которые ни разу не встречаются в массиве. Создать функцию для поиска элемента в массиве.

13. Известен рост участников университетской команды по баскетболу. Вывести фамилии самого низкого и самого высокого спортсменов. Считать, что состав команды составляет 12 человек. Поиск минимального и максимального роста оформить в виде функций.

13. Известен рост участников университетской команды по баскетболу. Вывести фамилии самого низкого и самого высокого спортсменов. Считать, что состав команды составляет 12 человек. Поиск минимального и максимального роста оформить в виде функций.

14. Известен возраст персонала предприятия. Вывести фамилии сотрудников мужского пола, которые относятся к категории предпенсионеров. Считать, что штат предприятия составляет 20 человек. Определение отношения к категории предпенсионеров оформить в виде функции; правила определения представлены в таблице.

ГОД	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028
Общеустановленный «новый» пенсионный возраст	61	62	63	64	65	65	65	65	65	65
Возраст отнесения мужчин к категории граждан предпенсионного возраста	56	57	58	59	60	60	60	60	60	60
Год рождения мужчин, которые относятся к категории граждан предпенсионного возраста исходя из общеустановленного «нового» пенсионного возраста	1959									
	1960	1960	1960							
	1961	1961	1961	1961	1961					
	1962	1962	1962	1962	1962	1962	1962			
	1963	1963	1963	1963	1963	1963	1963	1963	1963	
						1964	1964	1964	1964	1964
							1965	1965	1965	1965
								1966	1966	1966
									1967	1967
										1968

15. Известен возраст персонала предприятия. Вывести фамилии сотрудников женского пола, которые относятся к категории предпенсионеров. Считать, что штат предприятия составляет 20 человек. Определение отношения к категории предпенсионеров оформить в виде функции; правила определения представлены в таблице.

ГОД	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028
Общеустановленный «новый» пенсионный возраст	56	57	58	59	60	60	60	60	60	60
Возраст отнесения женщин к категории граждан предпенсионного возраста	51	52	53	54	55	55	55	55	55	55
Год рождения женщин, которые относятся к категории граждан предпенсионного возраста исходя из общеустановленного «нового» пенсионного возраста	1964									
	1965	1965	1965							
	1966	1966	1966	1966	1966					
	1967	1967	1967	1967	1967	1967	1967			
	1968	1968	1968	1968	1968	1968	1968	1968	1968	
						1969	1969	1969	1969	1969
							1970	1970	1970	1970
								1971	1971	1971
									1972	1972
										1973

Задание 2. Матрицы

Пример реализации

```
/* Пусть разреженная квадратная матрица задана в виде двумерного
массива.
Написать функцию, которая возвращает столбец с максимальной
суммой.*/
#include <iostream>
#include <algorithm>
#include <vector>

int get_max_sum_column_index(double** matrix, const int &columns,
const int &rows)
{
	int max_sum = -1;
	int max_sum_column_index = -1;
	for (size_t i = 0; i < columns; i++)
	{
		int sum = 0;
		for (size_t j = 0; j < rows; j++)
		{
			sum += matrix[j][i];
		}

		if (sum > max_sum)
		{
			max_sum = sum;
			max_sum_column_index = i;
		}
```

```
	}

	return max_sum_column_index;
}

int main()
{
	srand(0);

	const int rows		= 10;
	const int columns	= 10;

	double** matrix = new double* [rows];

	for (size_t i = 0; i < rows; i++)
	{
		matrix[i] = new double[columns];

		for (size_t j = 0; j < columns; j++)
		{
			matrix[i][j] = rand() % 50;
			std::cout << matrix[i][j] << "\t";
		}

		std::cout << std::endl;
	}

	std::cout << std::endl << "Max sum column index: " <<
get_max_sum_column_index(matrix, columns, rows);

	for (size_t i = 0; i < rows; i++)
		delete[] matrix[i];

	delete[] matrix;

	return 0;
}
```

Варианты

1. Пусть разреженные матрицы заданы в виде двумерных массивов. Написать функцию, которая возвращает сумму двух матриц.
2. Пусть разреженные матрицы заданы в виде двумерных массивов. Написать функцию, которая возвращает разность двух матриц.
3. Пусть разреженные матрицы заданы в виде двумерных массивов. Написать функцию, которая возвращает матрицу, умноженную на число k.
4. Пусть разреженные матрицы заданы в виде двумерных массивов. Написать функцию, которая возвращает произведение двух матриц.
5. Пусть разреженная квадратная матрица задана в виде двумерного массива. Транспонировать матрицу.

6. Пусть разреженная квадратная матрица задана в виде двумерного массива. Найти обратную матрицу.

7. Пусть разреженная квадратная матрица задана в виде двумерного массива. Написать функцию, которая возвращает матрицу, возведенную в степень k.

8. Пусть разреженная квадратная матрица задана в виде двумерного массива. Написать функцию, которая возвращает массив, состоящий из элементов главной диагонали.

9. Пусть разреженная квадратная матрица задана в виде двумерного массива. Написать функцию, которая возвращает массив, состоящий из элементов побочной диагонали.

10. Пусть разреженная квадратная матрица задана в виде двумерного массива. Написать функцию, которая возвращает строку с максимальной суммой.

11. Пусть разреженная квадратная матрица задана в виде двумерного массива. Написать функцию, которая возвращает столбец с максимальной суммой.

12. Даны два одномерных упорядоченных массива. Написать функцию, которая возвращает массив, объединяющий исходные с сохранением порядка как в первом массиве.

13. Даны два одномерных упорядоченных массива. Написать функцию, которая возвращает массив, объединяющий исходные с сохранением порядка как в более длинном массиве.

14. Дан одномерный массив. Написать функцию, которая возвращает массив, в котором удалены все компоненты, которые присутствуют более одного раза.

15. Пусть последовательность целых чисел задана в виде одномерного массива. Написать функцию, которая возвращает самую длинную возрастающую подпоследовательность.

Задание 3 .Шаблоны функций

Пример реализации

```
/* Написать шаблон функции Min_Array() для поиска минимального
элемента в одномерном массиве. Программа должна искать минимальный
элемент в целочисленном, вещественном, символьном массивах.
Глобальные переменные и константы не использовать */
#include <iostream>

using namespace std;
```

```
template <typename T> // создаем шаблон функции Min_Array. Мы используем шаблон функции когда не знаем какого типа дпнные

T Min_Array(const T* array, int count) // принимаем массив встроенного типа T(любой из возможный типов данных int, double и тд) и его размер
{
    T min = array[0]; // создаем переменную встроенного типа T

    // ищем минимум в массиве
    for (int i = 0; i < count; count++)
    {
        if (array[i] < min)
        {
            min = array[i];
        }
    }
    return min; // возвращаем найденное значение
}

int main()
{
    int array_1[5] = { 1, 2, 3, 4, 5 };

    double array_2[5] = { 0.5, 1.5, 2, 3, 5.7 };

    char array_3[5] = { 'a', 'b', 'c', 'd', 'e' };

    cout << Min_Array(array_1, 5);
    cout << Min_Array(array_2, 5);
    cout << Min_Array(array_3, 5);

    return 0;
}
```

Варианты

1. Написать шаблон функции Print_Array() для вывода одномерного массива на экран. Программа должна выводить на экран целочисленный, вещественный, символьный массивы, объявленные и инициализированные в тексте программы. Глобальные переменные не использовать.

2. Написать шаблон функции Max_Array() для поиска максимального элемента в одномерном массиве. Программа должна искать максимальный элемент в целочисленном, вещественном, символьном массивах. Глобальные переменные и константы не использовать.

3. Написать шаблон функции Sum_Array() для вычисления суммы элементов в одномерном массиве. Программа должна считать сумму для целочисленного и вещественного (float и double) массивов. Глобальные переменные и константы не использовать.

4. Написать шаблон функции Avg_Array() для вычисления среднего арифметического элементов одномерного массива. Программа должна находить среднее арифметическое целочисленного и вещественного (float и double) массивов. Глобальные переменные и константы не использовать.

5. Написать шаблон функции Sort_Array() для сортировки одномерного массива. Программа должна сортировать целочисленный, символьный и вещественный (float и double) массивы. Глобальные переменные и константы не использовать. Алгоритм сортировки выбрать на своё усмотрение.

6. Написать шаблон функции Sigma_Array() для определения среднеквадратического отклонения S одномерного массива. Программа должна работать для целочисленного и вещественного (float и double) массивов. Глобальные переменные и константы не использовать. Формула для расчёта:

$$S = \sqrt{S^2}$$, где

$$S^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$, где

N – количество элементов массива,

x_i – элемент массива,

$\bar{x}$ – среднее арифметическое,

7. Написать шаблон функции Sigma2_Array() для определения дисперсии S2 одномерного массива. Программа должна работать для целочисленного и вещественного (float и double) массивов. Глобальные переменные и константы не использовать. Формула для расчёта:

$$S^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$, где

N – количество элементов массива,

x_i – элемент массива,

$\bar{x}$ – среднее арифметическое,

8. Написать шаблон функции Diff_Array() для определения разницы между максимальным и минимальным элементами одномерного массива. Программа должна работать для целочисленного и вещественного (float и double) массивов. Глобальные переменные и константы не использовать.

9. Написать шаблон функции LMax_Array() для определения количества локальных максимумов одномерного массива. Локальным максимумом называется элемент массива, который больше своих соседей. Программа должна работать для целочисленного и вещественного

(float и double) массивов. Глобальные переменные и константы не использовать.

10. Написать шаблон функции LMin_Array() для определения количества локальных минимумов одномерного массива. Локальным минимумом называется элемент массива, который меньше своих соседей. Программа должна работать для целочисленного и вещественного (float и double) массивов. Глобальные переменные и константы не использовать.

11. Написать шаблон функции PSum_Array() для вычисления суммы положительных элементов в одномерном массиве. Программа должна считать сумму для целочисленного и вещественного (float и double) массивов. Глобальные переменные и константы не использовать.

12. Написать шаблон функции NSum_Array() для вычисления суммы отрицательных элементов в одномерном массиве. Программа должна считать сумму для целочисленного и вещественного (float и double) массивов. Глобальные переменные и константы не использовать.

13. Написать шаблон функции PAvg_Array() для вычисления среднего арифметического положительных элементов одномерного массива. Программа должна находить среднее арифметическое для целочисленного и вещественного (float и double) массивов. Глобальные переменные и константы не использовать.

14. Написать шаблон функции NAvg_Array() для вычисления среднего арифметического отрицательных элементов одномерного массива. Программа должна находить среднее арифметическое для целочисленного и вещественного (float и double) массивов. Глобальные переменные и константы не использовать.

15. Написать шаблон функции Zero_Array() для вычисления количества нулевых элементов одномерного массива. Программа должна находить количество для целочисленного и вещественного (float и double) массивов. Глобальные переменные и константы не использовать.

1.9. Динамические структуры данных

Задание 1. Стек

Пример реализации 1

```
/*Дан указатель P1 на вершину непустого стека.
Создать два новых стека, переместив в первый из них все элементы
исходного стека с четными значениями, а во второй – с нечетными
(элементы в новых стеках будут располагаться в порядке, обратном
исходному; один из этих стеков может оказаться пустым).
```

```
Вывести адреса вершин полученных стеков (для пустого стека вывести
nil). Операции выделения и освобождения памяти не использовать.*/

#include <stdio.h>
#include <malloc.h>

struct node {
    int val;
    struct node* next;
};

//разделение изначального стека на два стека
void stack_split(struct node** s1,
                 struct node** s2,
                 struct node** st){
    struct node* p = *st;
    for(*s1 = *s2 = NULL; p != NULL; ){
        if(p->val % 2){
            *s1 = p;
             p  = p->next;
             s1 = &(*s1)->next;
        } else {
            *s2 = p;
             p  = p->next;
             s2 = &(*s2)->next;
        }
    }
    *s1 = *s2 = *st = NULL;
}
//втолкнуть элемент в стек
int stack_push(struct node** st, int val){
    struct node* p = (struct node*)malloc(sizeof(struct node));
    if(p != NULL){
        p->val  = val;
        p->next = *st;
        *st     = p;
    }
    return (p != NULL);
}

//вытолкнуть элемент из стека
void stack_pop(struct node** st){
    struct node* t = *st;
    if(t != NULL){
        *st = (*st)->next;
        free(t);
    }
}
int main(void){
    int i;
    struct node* s1, *s2, *st = NULL;

    for(i = 0; i < 30; ++i)
```

```
        stack_push(&st, i);

    stack_split(&s1, &s2, &st);

    //адреса вершин новых стеков
    printf("addr ptr - s1: 0x%08X\n", s1);
    printf("addr ptr - s2: 0x%08X\n\n", s2);

    //вывести нечётные числа
    while(s1 != NULL){
        printf("%d ", s1->val);
        s1 = s1->next;
    }
    putchar('\n');

    //вывести чётные числа
    while(s2 != NULL){
        printf("%d ", s2->val);
        s2 = s2->next;
    }
    putchar('\n');
    return 0;
}
```

Пример реализации 2 (библиотека stack подключена)

```
/*Дан указатель P1 на вершину непустого стека.
Создать два новых стека, переместив в первый из них все элементы
исходного стека с четными значениями, во второй — с нечетными
(элементы в новых стеках будут располагаться в порядке, обратном
исходному; один из этих стеков может оказаться пустым). Вывести
адреса вершин полученных стеков (для пустого стека вывести nil).
Операции выделения и освобождения памяти не использовать.*/
#include <iostream>
#include <stack>
#include<ctime>// подключаем библиотеку для использования стека
using namespace std;

int main() {
    setlocale(LC_ALL, "rus");
    stack <int> steck;  // создаем стек
    int n;

    cout << "Введите число элементов в стеке: " << endl; //
    cin >> n;
    srand(time(0));//
    for(int i=0; i<n;i++ ){
        int a;
        a = rand()%10 ;
        steck.push(a);  // добавляем введенные числа
        cout << a << ' ';
    }
// проверяем пуст ли стек (нет)
    if (steck.empty()) cout << "Стек не пуст";
```

```
    std::stack<int> s1, s2;
    int temp;
    while (!steck.empty())
    {
        temp = steck.top();
        steck.pop();
        if (temp % 2==0) {
            s1.push(temp);
        }
        else {
            s2.push(temp);
        }
    }
    cout << endl;
    if (s1.empty())
    {
        cout << "Адрес стека с четными чисами - nill"<<endl;
    }
    else {
        printf("Адрес стека с четными числами - s1: 0x%08X\n",
s1);
    }
    if (s2.empty())
    {
        cout << "Адрсес стека с нечетными числами - nill";
    }
    else {
        printf("Адрес стека с нечетными числами - s2: 0x%08X\n\n",
s2);
    }
    }
```

Варианты

1. Дан адрес P_1 записи типа TNode, содержащей поле Data (целого типа) и поле Next (типа PNode — указателя на TNode). Эта запись связана полем Next со следующей записью того же типа. Вывести значения полей Data обеих записей, а также адрес P_2 следующей записи.

2º. Дан адрес P_1 записи типа TNode. Эта запись связана полем Next со следующей записью того же типа, она, в свою очередь, — со следующей, и так далее до записи, поле Next которой равно nil (таким образом, возникает *цепочка* связанных записей). Вывести значения полей Data для всех элементов цепочки, *длину цепочки* (то есть число ее элементов) и адрес ее последнего элемента.

3º. Дано число D и указатель P_1 на вершину непустого стека. Добавить элемент со значением D в стек и вывести адрес P_2 новой вершины стека.

4. Дано число N (> 0) и набор из N чисел. Создать стек, содержащий исходные числа (последнее число будет вершиной стека), и вывести указатель на его вершину.

5º. Дан указатель P_1 на вершину непустого стека. Извлечь из стека первый

(верхний) элемент и вывести его значение D, а также адрес P_2 новой вершины стека. Если после извлечения элемента стек окажется пустым, то положить P_2 = nil. После извлечения элемента из стека освободить память, занимаемую этим элементом.

6. Дан указатель P_1 на вершину стека, содержащего не менее десяти элементов. Извлечь из стека первые девять элементов и вывести их значения. Вывести также адрес новой вершины стека. После извлечения элементов из стека освобождать память, которую они занимали.

7. Дан указатель P_1 на вершину стека (если стек пуст, то P_1 = nil). Извлечь из стека все элементы и вывести их значения. Вывести также количество извлеченных элементов N (для пустого стека вывести 0). После извлечения элементов из стека освобождать память, которую они занимали.

8º. Даны указатели P_1 и P_2 на вершины двух непустых стеков. Переместить все элементы из первого стека во второй (в результате элементы первого стека будут располагаться во втором стеке в порядке, обратном исходному) и вывести адрес новой вершины второго стека. Операции выделения и освобождения памяти не использовать.

9. Даны указатели P_1 и P_2 на вершины двух непустых стеков. Перемещать элементы из первого стека во второй, пока значение вершины первого стека не станет четным (перемещенные элементы первого стека будут располагаться во втором стеке в порядке, обратном исходному). Если в первом стеке нет элементов с четными значениями, то переместить из первого стека во второй все элементы. Вывести адреса новых вершин первого и второго стека (если первый стек окажется пустым, то вывести для него константу nil). Операции выделения и освобождения памяти не использовать.

10º. Дан указатель P_1 на вершину непустого стека. Создать два новых стека, переместив в первый из них все элементы исходного стека с кратными трём значениями, а во второй — с не кратными трём (элементы в новых стеках будут располагаться в порядке, обратном исходному; один из этих стеков может оказаться пустым). Вывести адреса вершин полученных стеков (для пустого стека вывести nil). Операции выделения и освобождения памяти не использовать.

11º. Дан указатель P_1 на вершину стека (если стек пуст, то P_1 = nil). Также дано число N (> 0) и набор из N чисел. Описать тип TStack — запись с одним полем Top типа PNode (поле указывает на *вершину стека*) — и процедуру Push(S, D), которая добавляет в стек S новый элемент со значением D (S — входной и выходной параметр типа TStack, D — входной па раметр целого типа). С помощью процедуры Push добавить в исходный стек данный набор чисел (последнее число будет вершиной стека) и вывести адрес новой вершины стека.

12. Дан указатель P_1 на вершину стека, содержащего не менее пяти элементов. Используя тип TStack, описать функцию Pop(S) целого типа, которая извлекает из стека S первый (верхний) элемент, возвращает его

значение и освобождает память, которую занимал извлеченный элемент (*S* — входной и выходной параметр типа TStack). С помощью функции Pop извлечь из исходного стека пять элементов и вывести их значения. Вывести также указатель на новую вершину стека (если результирующий стек окажется пустым, то этот указатель должен быть равен nil).

13. Дан указатель P_1 на вершину стека. Используя тип TStack (описать функции StackIsEmpty(*S*) логического типа (возвращает True, если стек *S* пуст, и False в противном случае) и Peek(*S*) целого типа (возвращает значение вершины непустого стека *S*, не удаляя ее из стека). В обеих функциях переменная *S* является входным параметром типа TStack. С помощью этих функций, а также функции Pop извлечь из исходного стека пять элементов (или все содержащиеся в нем элементы, если их менее пяти) и вывести их значения. Вывести также значение функции StackIsEmpty для результирующего стека и, если результирующий стек не является пустым, значение и адрес его новой вершины.

14. Дан указатель P_1 на вершину непустого стека. Написать функцию, которая вычисляет сумму элементов стека. Вывести на экран элементы стека и их сумму.

15. Дан указатель P_1 на вершину непустого стека. Написать функцию, которая подсчитывает количество ненулевых элементов в стеке. Вывести на экран элементы стека и результат работы функции.

Задание 2. Очередь

Пример реализации 1

```
/*Даны указатели P1 и P2 на начало и конец очереди.
Используя тип TQueue – запись с двумя полями типа PNode: Head
(начало очереди) и Tail (конец очереди) – описать функцию
QueueIsEmpty(Q) логического типа,которая возвращает True, если
очередь Q пуста, и False в противном случае (Q – входной параметр
типа TQueue).
Используя эту функцию для проверки состояния очереди, а также
функцию Dequeue, извлечь из исходной очереди пять начальных
элементов (или все содержащиеся в ней элементы,
если их менее пяти) и вывести их значения. Вывести также значение
функции QueueIsEmpty для полученной очереди и новые адреса ее
начала и конца.*/
#include <iostream>
#include <math.h>

using namespace std;

struct Node {
    int data;
    Node *Next;
};
typedef Node *PNode;
struct T{
```

```
    PNode Head;
    PNode Tail;
};
typedef T *TQueue;

bool QueueIsEmpty(TQueue Q)
{
    if (Q->Tail != NULL) return false;
    return true;
}

int pop(TQueue Q) // функция Decueue
{
    int temple;
    if (Q->Head != NULL){
        temple = Q->Head->data;
        if (Q->Head == Q->Tail) Q->Tail = NULL;
        Q->Head = Q->Head->Next;
        return temple;
    }
    return 0;
}

void push(TQueue Q,int n) // создание корня и добавление элементов
{
    PNode newNode = new Node;
    newNode->data = n;
    newNode->Next = NULL;

    bool flag = false;
    if (Q->Head == Q->Tail and Q->Tail != NULL) flag = true;

    if (Q->Tail != NULL) {
        Q->Tail->Next = newNode;
    }
    Q->Tail = newNode;
    if (flag) Q->Head->Next = Q->Tail;
    if (Q->Head == NULL) Q->Head = Q->Tail;
}

int main()
{
    TQueue Q = new T;
    Q->Head = NULL;
    Q->Tail = NULL;

    int i = 2;
    while (i != 12){
        push(Q,ceil(pow(i,2.5)));
        i++;
    }
    int count = 5;
    while (count > 0 and !QueueIsEmpty(Q)){
```

```
        cout<<pop(Q)<<endl;
        count--;
    }
    cout<<"QueueIsEmpty? - ";
    QueueIsEmpty(Q) ? cout<<"True": cout<<"False";
    cout<<endl<<"Address Head: "<<Q->Head<<endl<<"Address Tail:
"<<Q->Tail;
    return 0;
}
```

Пример реализации 2 (библиотека queue подключена)

```
*/Дан набор из 10 чисел. Создать две очереди: первая должна
содержать все нечетные, а вторая — все четные числа из исходного
набора (по рядок чисел в каждой очереди должен совпадать с
порядком чисел в исходном наборе). Вывести указатели на начало и
конец первой, а затем второй очереди (одна из очередей может
оказаться пустой; в этом случае вывести для нее две константы
nil).*/
#include <iostream>
#include <queue>  // подключили библиотеку queue

using namespace std;

int main() {
  setlocale(LC_ALL,"rus");
  queue <int> q;  // создали очередь q
  queue <int> p;  // создали очередь q

  cout << "Пользователь, пожалуйста введите 10 чисел: " << endl;

  for (int h = 0; h < 10; h++) {
    int a;

    cin >> a;

    if (a%2!=0)
    {
    q.push(a);  // добавляем в очередь элементы
    }
    else
    {
    p.push(a);  // добавляем в очередь элементы
    }
  }
   if (!q.empty())
   {
   cout << "Самый первый элемент в очереди в 1 очереди: " <<
&q.front() << endl;
   cout << "Самый последний элемент в очереди в 1 очереди: " <<
&q.back() << endl;
   }
   else
   {
```

```
   cout<<"nil"<<endl;
   cout<<"nil"<<endl;
   }
   if (!p.empty())
   {
    cout << "Самый первый элемент в очереди во 2 очереди: " <<
&p.front() << endl;
    cout << "Самый последний элемент в очереди во 2 очереди: " <<
&p.back() << endl;
   }
   else
   {
    cout<<"nil"<<endl;
    cout<<"nil"<<endl;
   }

  return 0;
                                                    }
```

Варианты

1. Дан набор из 10 чисел. Создать очередь, содержащую данные числа в указанном порядке (первое число будет размещаться в начале очереди, последнее — в конце), и вывести указатели P_1 и P_2 на начало и конец очереди.

2. Дан набор из 10 чисел. Создать две очереди: первая должна содержать числа из исходного набора с нечетными номерами (1, 3, …, 9), а вторая — с четными (2, 4, …, 10); порядок чисел в каждой очереди должен совпадать с порядком чисел в исходном наборе. Вывести указатели на начало и конец первой, а затем второй очереди.

3. Дан набор из 10 чисел. Создать две очереди: первая должна содержать все четные, а вторая — все нечетные числа из исходного набора (порядок чисел в каждой очереди должен совпадать с порядком чисел в исходном наборе). Вывести указатели на начало и конец первой, а затем второй очереди (одна из очередей может оказаться пустой; в этом случае вы вести для нее две константы nil).

4. Дано число D и указатели P_1 и P_2 на начало и конец очереди (если очередь является пустой, то $P_1 = P_2 =$ nil). Добавить элемент со значением D в конец очереди и вывести новые адреса начала и конца очереди.

5. Дано число D и указатели P_1 и P_2 на начало и конец очереди, со держащей не менее двух элементов. Добавить элемент со значением D в конец очереди и извлечь из очереди первый (начальный) элемент. Вывести значение извлеченного элемента и новые адреса начала и конца очереди. После извлечения элемента из очереди освободить память, занимаемую этим элементом.

6. Дано число N (> 0) и указатели P_1 и P_2 на начало и конец непустой очереди. Извлечь из очереди N начальных элементов и вывести их значения (если очередь содержит менее N элементов, то извлечь все ее элементы). Вывести также новые адреса начала и конца очереди (для пустой оче реди

дважды вывести nil). После извлечения элементов из очереди осво бождать память, которую они занимали.

7. Даны указатели P_1 и P_2 на начало и конец непустой очереди. Из влекать из очереди элементы, пока значение начального элемента очереди не станет четным, и выводить значения извлеченных элементов (если очередь не содержит элементов с четными значениями, то извлечь все ее эле менты). Вывести также новые адреса начала и конца очереди (для пустой очереди дважды вывести nil). После извлечения элементов из очереди освобождать память, которую они занимали.

8. Даны две очереди; адреса начала и конца первой равны P_1 и P_2, а второй — P_3 и P_4 (если очередь является пустой, то соответствующие адреса равны nil). Переместить все элементы первой очереди (в порядке от начала к концу) в конец второй очереди и вывести новые адреса начала и конца второй очереди. Операции выделения и освобождения памяти не использовать.

9. Дано число N (> 0) и две непустые очереди; адреса начала и конца первой равны P_1 и P_2, а второй — P_3 и P_4. Переместить N начальных эле ментов первой очереди в конец второй очереди. Если первая очередь со держит менее N элементов, то переместить из первой очереди во вторую все элементы. Вывести новые адреса начала и конца первой, а затем второй очереди (для пустой очереди дважды вывести nil). Операции выделения и освобождения памяти не использовать.

10. Даны две непустые очереди; адреса начала и конца первой равны P_1 и P_2, а второй — P_3 и P_4. Перемещать элементы из начала первой очереди в конец второй, пока значение начального элемента первой очереди не станет четным (если первая очередь не содержит четных элементов, то переместить из первой очереди во вторую все элементы). Вывести новые адреса начала и конца первой, а затем второй очереди (для пустой очереди дважды вывести nil). Операции выделения и освобождения памяти не использовать.

11. Даны две непустые очереди; адреса начала и конца первой равны P_1 и P_2, а второй — P_3 и P_4. Очереди содержат одинаковое количество элементов. Объединить очереди в одну, в которой элементы исходных очередей чередуются (начиная с первого элемента первой очереди). Вывести указатели на начало и конец полученной очереди. Операции выделения и освобождения памяти не использовать.

12. Даны две непустые очереди; адреса начала и конца первой равны P_1 и P_2, а второй — P_3 и P_4. Элементы каждой из очередей упорядочены по возрастанию (в направлении от начала очереди к концу). Объединить очереди в одну с сохранением упорядоченности элементов. Вывести указатели на начало и конец полученной очереди. Операции выделения и освобождения памяти не использовать, поля Data не изменять.

13. Даны указатели P_1 и P_2 на начало и конец очереди (если очередь является пустой, то $P_1 = P_2$ = nil). Также дано число N (> 0) и набор из N чисел. Описать тип TQueue — запись с двумя полями типа PNode: Head (*начало*

очереди) и Tail (*конец очереди*) — и процедуру Enqueue(*Q*, *D*), ко торая добавляет в конец очереди *Q* новый элемент со значением *D* (*Q* — входной и выходной параметр типа TQueue, *D* — входной параметр целого типа). С помощью процедуры Enqueue добавить в исходную очередь данный набор чисел и вывести новые адреса ее начала и конца.

14. Даны указатели P_1 и P_2 на начало и конец очереди, содержащей не менее пяти элементов. Используя тип TQueue— запись с двумя полями типа PNode: Head (*начало очереди*) и Tail (*конец очереди*) — описать функцию Dequeue(*Q*) целого типа, которая извлекает из очереди первый (начальный) элемент, возвращает его значение и освобождает память, занимаемую извлеченным элементом (*Q* — входной и выходной параметр типа TQueue). С помощью функции Dequeue извлечь из исходной очереди пять начальных элементов и вывести их значения. Вывести также адреса начала и конца результирующей очереди (если очередь окажется пустой, то эти адреса должны быть равны nil).

15. Даны указатели P_1 и P_2 на начало и конец очереди. Используя тип TQueue — запись с двумя полями типа PNode: Head (*начало очереди*) и Tail (*конец очереди*) — описать функцию QueueIsEmpty(*Q*) логического типа, которая возвращает True, если очередь *Q* пуста, и False в противном случае (*Q* — входной параметр типа TQueue). Используя эту функцию для проверки состояния очереди, а также функцию Dequeue, извлечь из исходной очереди пять начальных элементов (или все содержащиеся в ней элементы, если их менее пяти) и вывести их значения. Вывести также значение функции QueueIsEmpty для полученной очереди и новые адреса ее начала и конца.

Задание 3. Двусвязный список

Пример реализации

```
/*Дан указатель P1 на первый элемент непустого двусвязного списка. Удалить из списка все элементы с нечетными значениями и вывести указатель на первый элемент преобразованного списка (если в результате удаления элементов список окажется пустым, то вывести nil). После удаления элементов из списка освобождать память, которую они занимали.*/
#include <iostream>

using namespace std;

struct Node {
    int data;
    Node *next;
    Node *back;
};
typedef Node *PNode;
PNode P1;

void pop(PNode lst)
{
```

```
    PNode prev, nextt;
    prev = lst->back;
    nextt = lst->next;
    if (prev != NULL) prev->next = lst->next;
    if (nextt != NULL) nextt->back = lst->back;
    if (P1 == lst) P1 = nextt;
    free(lst);
}

PNode push(PNode lst, int number)
{
    PNode temp, p;
    temp = (PNode)malloc(sizeof(Node));
    p = lst->next;
    lst->next = temp;
    temp->data = number;
    temp->next = p;
    temp->back = lst;
    if (p != NULL) p->back = temp;
    return temp;
}

PNode CreateQueue(int a)
{
    PNode lst;
    lst = (PNode)malloc(sizeof(struct Node));
    lst->data = a;
    lst->next = NULL;
    lst->back = NULL;
    P1 = lst;
    return(lst);
}

int main()
{
    PNode temp;
    temp = CreateQueue(0);
    int i = 1;
    while (i<20){
        temp = push(temp,i);
        i++;
    }
    PNode temp1;
    temp = P1;
    while (temp != NULL){
        if (temp->data%2 == 1) {
            temp1 = temp;
            temp = temp->next;
            pop(temp1);
        }
        else temp = temp->next;
    }
    if (P1 != NULL) cout<<P1;
```

```
    else cout<<"nil";

    return 0;
}
```

Пример реализации 2 (библиотека deque подключена)

```
/*Дан указатель P0 на один из элементов непустого двусвязного
списка. Вывести число N — количество элементов в списке, а также
указатели P1 и P2 на первый и последний элементы списка.*/
#include <iostream>
#include <deque>
#include <stdexcept>
using namespace std;

int main()
{
    std::deque<int> num;
    int a, n;
    a=rand();
    n=rand()%100;
    for (int i=0;i<n;i++)
    {
        a++;
        num.push_front(a);
    }
    cout<<"количество чисел в списке "<<n<<endl;
    cout<<&num.front()<<endl;
    cout<<&num.back()<<endl;

    return 0;
}
```

Варианты

1. Дан адрес P_2 записи типа TNode, содержащей поле Data (целого типа) и поля Prev и Next (типа PNode — указателя на TNode). Эта запись связана полями Prev и Next соответственно с предыдущей и последующей записью того же типа. Вывести значения полей Data предыдущей и последующей записи, а также адреса P_1 и P_3 предыдущей и последующей записи.

2º. Дан указатель P_1 на начало непустой цепочки элементов записей типа TNode, связанных между собой с помощью поля Next. Используя по ле Prev записи TNode, преобразовать исходную (*односвязную*) цепочку в *двусвязную*, в которой каждый элемент связан не только с последующим элементом (с помощью поля Next), но и с предыдущим (с помощью поля Prev). Поле Prev первого элемента положить равным nil. Вывести указатель на последний элемент преобразованной цепочки.

3. Дан указатель P_0 на один из элементов непустого двусвязного списка. Вывести число N — количество элементов в списке, а также указатели P_1 и P_2 на первый и последний элементы списка.

4. Даны числа D_1 и D_2 и указатель P_0 на один из элементов непустого

двусвязного списка. Добавить в начало списка новый элемент со значением D_1, а в конец — новый элемент со значением D_2. Вывести адреса первого и последнего элементов полученного списка.

5. Дано число D и указатель P_0 на один из элементов непустого двусвязного списка. Вставить перед данным элементом списка новый элемент со значением D и вывести указатель на добавленный элемент списка.

6. Дано число D и указатель P_0 на один из элементов непустого двусвязного списка. Вставить после данного элемента списка новый элемент со значением D и вывести указатель на добавленный элемент списка.

7. Даны указатели P_1 и P_2 на первый и последний элементы двусвязного списка, содержащего не менее двух элементов. Продублировать в списке первый и последний элементы (новые элементы добавлять перед существующими элементами с такими же значениями) и вывести указатель на первый элемент преобразованного списка.

8. Даны указатели P_1 и P_2 на первый и последний элементы двусвязного списка, содержащего не менее двух элементов. Продублировать в списке первый и последний элементы (новые элементы добавлять после существующих элементов с такими же значениями) и вывести указатель на последний элемент преобразованного списка.

9. Дан указатель P_1 на первый элемент непустого двусвязного списка. Продублировать в списке все элементы с нечетными номерами (новые элементы добавлять перед существующими элементами с такими же значениями) и вывести указатель на первый элемент преобразованного списка.

10. Дан указатель P_1 на первый элемент непустого двусвязного списка. Продублировать в списке все элементы с нечетными номерами (новые элементы добавлять после существующих элементов с такими же значениями) и вывести указатель на последний элемент преобразованного списка.

11. Дан указатель P_1 на первый элемент непустого двусвязного списка. Продублировать в списке все элементы с нечетными значениями (новые элементы добавлять перед существующими элементами с такими же зна чениями) и вывести указатель на первый элемент преобразованного спи ска.

12. Дан указатель P_1 на первый элемент непустого двусвязного списка. Продублировать в списке все элементы с нечетными значениями (новые элементы добавлять после существующих элементов с такими же значениями) и вывести указатель на последний элемент преобразованного списка.

13. Дан указатель P_0 на один из элементов непустого двусвязного спи ска. Удалить из списка данный элемент и вывести два указателя: на эле мент, предшествующий удаленному, и на элемент, следующий за удаленным (один или оба этих элемента могут отсутствовать; для отсутствующих элементов выводить nil). После удаления элемента из списка освободить память, занимаемую этим элементом.

14. Дан указатель P_1 на первый элемент двусвязного списка, содержа щего не менее двух элементов. Удалить из списка все элементы с нечетными

номерами и вывести указатель на первый элемент преобразованного списка. После удаления элементов из списка освобождать память, которую они занимали.

15. Дан указатель P_1 на первый элемент непустого двусвязного списка. Удалить из списка все элементы с четными значениями и вывести указатель на первый элемент преобразованного списка (если в результате удаления элементов список окажется пустым, то вывести nil). После удаления элементов из списка освобождать память, которую они занимали.

2. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

2.1. Классы. Конструкторы и деструкторы

В ходе лабораторной работы необходимо описать класс, для которого предусмотреть методы установки свойств (set…) и получения информации о текущем состоянии(get). Для каждого класса в обязательном порядке необходимо создать конструктор с параметрами и деструктор.

Пример реализации

```
    /*Создать класс для объектов-векторов, задаваемых координатами
концов в двумерном пространстве. Создать методы для выполнения
операций сложения и вычитания векторов с получением нового вектора
(суммы или разности), вычисления скалярного произведения двух
векторов, длины вектора, косинуса угла между векторами.*/
# include <iostream>
# include <math.h>
using namespace std;

class AB
{
 private:
 int x;
 int y;
 public:
 AB()
 {
  x = 0;
  y = 0;
 }

 void setAB(int a, int b)
 {
  x=a;
  y=b;
 }

 void getAB()
 {
  cout << "x = " << x << endl << endl;
  cout << "y = " << y << endl << endl;
 }
 void summ(AB *obj1, AB *obj2)
{
 cout<<"кординаты вектора суммы двух векторов"<<endl;
 x = obj1->x + obj2->x;
 y = obj1->y + obj2->y;
}
void rz(AB *obj1, AB *obj2)
{
 cout<<"кординаты вектора разности двух векторов"<<endl;
 x = obj1->x - obj2->x;
```

```
 y = obj1->y - obj2->y;
}
int sklr(AB *obj1, AB *obj2)
{
 int d;
 d = obj1->x * obj2->x + obj1->y * obj2->y;
 cout<<"скалярное произведение равно: "<<d<<endl;
}
int dl1(AB *obj1)
{
 float n;
 n = sqrt(obj1->x*obj1->x+obj1->y*obj1->y);
 cout<<"длина 1 вектора равна: "<<n<<endl;
}
int dl2(AB *obj2)
{
 float m;
 m = sqrt(obj2->x*obj2->x+obj2->y*obj2->y);
 cout<<"длина 2 вектора равна: "<<m<<endl;
}

int cosin(AB *obj1, AB *obj2)
{
 float co;
 co = (obj1->x * obj2->x + obj1->y * obj2->y)/(sqrt(obj1->x*obj1-
>x+obj1->y*obj1->y+)*sqrt(obj2->x*obj2->x+obj2->y*obj2->y));
 cout<<"косинус угла между векторами равен: "<<co<<endl;
}
 ~AB() { }
};

int main()
{
 setlocale(LC_ALL, "rus");
 int a,b,c;
 AB obj1;
 cout<<"Введите координаты 1го вектора:"<<endl;
 cout<<"x: ";
 cin>>a;
 cout<<"y: ";
 cin>>b;
obj1.setAB(a,b);
AB obj2;
 cout<<"Введите координаты 2го вектора:"<<endl;
 cout<<"x: ";
 cin>>a;
 cout<<"y: ";
 cin>>b;
obj2.setAB(a,b);
AB obj3;
obj3.summ(&obj1, &obj2);
obj3.getAB();
AB obj4;
```

```
obj4.rz(&obj1, &obj2);
obj4.getAB();
AB obj5;
obj5.sklr(&obj1, &obj2);
AB obj6;
obj6.dl1(&obj1);
AB obj7;
obj7.dl2(&obj2);
AB obj8;
obj8.cosin(&obj1, &obj2);
return 0;
}
```

Варианты

1. Создать класс, содержащий информацию о почтовом адресе организации. Предусмотреть возможность раздельного изменения составных частей адреса, создания и уничтожения объектов этого класса.

2. Создать класс для представления комплексных чисел. Создать методы для выполнения операций сложения, вычитания и умножения комплексных чисел.

3. Создать класс для объектов-векторов, задаваемых координатами концов в трехмерном пространстве. Создать методы для выполнения операций сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами.

4. Создать класс прямоугольников со сторонами, параллельными осям координат. Предусмотреть возможность перемещения прямоугольников на плоскости, изменение размеров, построение наименьшего прямоугольника, содержащего два заданных прямоугольника, являющегося общей частью (пересечением) двух прямоугольников.

5. Создать класс многочленов от одной переменной, задаваемых степенью многочлена и массивом коэффициентов. Предусмотреть методы для вычисления значения многочлена для заданного аргумента, операции сложения, вычитания и умножения многочленов с получением нового объекта-многочлена, вывод на экран описания многочлена.

6. Создать класс, обеспечивающий представление матрицы произвольного размера с возможностью изменения числа строк и столбцов, вывода на экран подматрицы любого размера и всей матрицы.

7. Создать класс для хранения календарных дат. Обеспечить возможность работы с датами в различных форматах, изменения даты на заданное количество дней. Стандартные функции и типы C для работы с датами не использовать.

8. Создать класс для хранения одномерных целочисленных массивов. Обеспечить возможность задания количества элементов и базовой индексации. Запрограммировать методы поиска элементов и сортировки.

9. Создать класс для хранения строк. Запрограммировать методы поиска подстроки, копирования, замены и удаления заданной подстроки, определения длины строки.

10. Создать класс для хранения обыкновенных дробей. Запрограммировать метод сокращения дроби.

11. Создать класс "Студент" для хранения персональных данных данных студентов. Запрограммировать методы ввода и вывода данных о студенте, перевода его на следующий курс и отчисления.

12. Создать класс «Колода карт», содержащий представляет собой массив структур, где первое поле — масть, второе — значимость. Создать методы для вывода текущего состояния колоды и размещения карт в производном порядке.

13. Создать класс «Абонент телефонной сети», который содержит следующие данные: 1) фамилия, Имя, Отчество, 2) улица, 3) дом, 4) квартира, 5) тел. номер, 6) тариф. Разработать методы для ввода и вывода данных об абоненте и смены тарифа.

14. Создать класс «Склад», где хранятся объекты типа «товар», для которых известно: 1) артикул, 2) количество, 3) цена. Предусмотреть методы для ввода и вывода данных, а также изменения количества и цены.

15. Создать класс «Поздравительная открытка», содержащих данные об адресате, текст поздравления, событие и его дату. Предусмотреть методы для ввода и вывода данных, а также изменения ФИО и события.

Полезные ссылки:

1. Липман С. Лажойе Ж. C++ для начинающих, главы 13, 14: https://cpp.com.ru/lippman/index.html

2. Уроки по C++. Конструкторы: Режим доступа: https://ravesli.com/urok-116-konstruktory/#toc-0

2.2. Классы. Дружественные функции

В данной работе требуется изменить описание класса таким образом, чтобы ввод и вывод данных осуществлялся с помощью дружественных функций.

Пример реализации

```
    /*Создать класс для объектов-векторов, задаваемых координатами концов в двумерном пространстве. Создать методы для выполнения операций сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами.*/
# include <iostream>
# include <math.h>
using namespace std;

class AB
{
 private:
 int x;
 int y;
 public:
 AB()
```

```
 {
  x = 0;
  y = 0;
 }
 void summ(AB *obj1, AB *obj2)
{
 cout<<"кординаты вектора суммы двух векторов"<<endl;
 x = obj1->x + obj2->x;
 y = obj1->y + obj2->y;
}
void rz(AB *obj1, AB *obj2)
{
 cout<<"кординаты вектора разности двух векторов"<<endl;
 x = obj1->x - obj2->x;
 y = obj1->y - obj2->y;
}
int sklr(AB *obj1, AB *obj2)
{
 int d;
 d = obj1->x * obj2->x + obj1->y * obj2->y;
 cout<<"скалярное произведение равно: "<<d<<endl;
}
int dl1(AB *obj1)
{
 float n;
 n = sqrt(obj1->x*obj1->x+obj1->y*obj1->y);
 cout<<"длина 1 вектора равна: "<<n<<endl;
}
int dl2(AB *obj2)
{
 float m;
 m = sqrt(obj2->x*obj2->x+obj2->y*obj2->y);
 cout<<"длина 2 вектора равна: "<<m<<endl;
}
int cosin(AB *obj1, AB *obj2)
{
 float co;
 co = (obj1->x * obj2->x + obj1->y * obj2->y+)/(sqrt(obj1->x*obj1-
>x+obj1->y*obj1->y)*sqrt(obj2->x*obj2->x+obj2->y*obj2->y));
 cout<<"косинус угла между векторами равен: "<<co<<endl;
}
 ~AB() { }
 friend void input (AB &ab);
 friend void output (AB &ab);
};
void input (AB &ab)
 {
  cout << " кординаты: "<<endl;
  cout << " x= ";
  cin>>ab.x;
  cout << " y= ";
  cin>>ab.y;
 }
```

```
 void output (AB &ab)
 {
  cout << " x= " << ab.x << endl;
  cout << " y= " << ab.y << endl;
}

int main()
{
  setlocale(LC_ALL, "rus");
 int a,b;
AB obj1;
 input(obj1);
AB obj2;
 input(obj2);
AB obj3;
obj3.summ(&obj1, &obj2);
 output(obj3);
AB obj4;
obj4.rz(&obj1, &obj2);
 output(obj4);
AB obj5;
obj5.sklr(&obj1, &obj2);
AB obj6;
obj6.dl1(&obj1);
AB obj7;
obj7.dl2(&obj2);
AB obj8;
obj8.cosin(&obj1, &obj2);
return 0;
}
```

Варианты

1. Создать класс, содержащий информацию о почтовом адресе организации. Предусмотреть возможность раздельного изменения составных частей адреса, создания и уничтожения объектов этого класса. Реализовать ввод и вывод почтового адреса с помощью дружественных функций.

2. Создать класс для представления комплексных чисел. Создать методы для выполнения операций сложения, вычитания и умножения комплексных чисел. Реализовать ввод и вывод комплексного числа с помощью дружественных функций.

3. Создать класс для объектов-векторов, задаваемых координатами концов в трехмерном пространстве. Создать методы для выполнения операций сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами. Реализовать ввод и вывод вектора с помощью дружественных функций.

4. Создать класс прямоугольников со сторонами, параллельными осям координат. Предусмотреть возможность перемещения прямоугольников на плоскости, изменение размеров, построение наименьшего прямоугольника, содержащего два заданных прямоугольника, являющегося общей частью

(пересечением) двух прямоугольников. Реализовать ввод координат и вывод прямоугольника с помощью дружественных функций.

5. Создать класс многочленов от одной переменной, задаваемых степенью многочлена и массивом коэффициентов. Предусмотреть методы для вычисления значения многочлена для заданного аргумента, операции сложения, вычитания и умножения многочленов с получением нового объекта-многочлена. Реализовать ввод коэффициентов и вывод на экран описания многочлена с помощью дружественных функций.

6. Создать класс, обеспечивающий представление матрицы произвольного размера с возможностью изменения числа строк и столбцов, вывода на экран подматрицы любого размера и всей матрицы. Реализовать ввод и вывод матрицы с помощью дружественных функций.

7. Создать класс для хранения календарных дат. Обеспечить возможность работы с датами в различных форматах, изменения даты на заданное количество дней. Стандартные функции и типы С для работы с датами не использовать. Реализовать ввод и вывод даты с помощью дружественных функций.

8. Создать класс для хранения одномерных целочисленных массивов. Обеспечить возможность задания количества элементов и базовой индексации. Запрограммировать методы поиска элементов и сортировки. Реализовать ввод и вывод массива с помощью дружественных функций.

9. Создать класс для хранения строк. Запрограммировать методы поиска подстроки, копирования, замены и удаления заданной подстроки, определения длины строки. Реализовать ввод и вывод строки с помощью дружественных функций.

10. Создать класс для хранения обыкновенных дробей. Запрограммировать метод сокращения дроби. Реализовать ввод и вывод дроби с помощью дружественных функций.

11. Создать класс "Студент" для хранения персональных данных данных студентов. Запрограммировать методы ввода и вывода данных о студенте, перевода его на следующий курс и отчисления. Реализовать ввод и вывод данных о студенте с помощью дружественных функций.

12. Создать класс «Колода карт», содержащий представляет собой массив структур, где первое поле — масть, второе — значимость. Реализовать ввод и вывод текущего состояния колоды с помощью дружественных функций.

13. Создать класс «Абонент телефонной сети», который содержит следующие данные: 1) фамилия, Имя, Отчество, 2) улица, 3) дом, 4) квартира, 5) тел. номер, 6) тариф. . Реализовать ввод и вывод данных об абоненте с помощью дружественных функций.

14. Создать класс «Склад», где хранятся объекты типа «товар», для которых известно: 1) артикул, 2) количество, 3) цена. Реализовать ввод и вывод данных о товаре с помощью дружественных функций.

15. Создать класс «Поздравительная открытка», содержащих данные об адресате, текст поздравления, событие и его дату. Реализовать ввод данных и вывод поздравления с помощью дружественных функций.

Полезные ссылки

1. Уроки по C++. Дружественные функции и классы. Режим доступа: https://ravesli.com/urok-126-druzhestvennye-funktsii-i-klassy/

2.3. Классы. Простое наследование

Пример реализации

```
    /*Разработать программу с использованием наследования классов, реализующую классы: массив; стек; очередь. Используя виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран количество элементов и добавьте элемент.*/

#include <iostream>
#include <iomanip>
using namespace std;

class Massive
{
protected:
	int size;
	int* mas;
public:
	Massive(int a)
	{
		size = a;
		mas = new int[size];
		cout << ""<<endl;
		for (int i = 0; i < size; i++)
		{
			cout <<"mas[" <<i<<"]: " ;
			cin >> mas[i];
		}
		cout << endl;
	}
	virtual void Print()
	{
		cout << "Massive:" << endl;
		for (int i=0;i<size;i++)
		{
			cout << setfill(' ') << setw(3) << mas[i];
		}
		cout << endl;
	}
	virtual void AddElement(int a)
	{
		int* b = new int[size + 1];
		for (int i = 0; i < size; i++)
		{
			b[i]=mas[i];
		}
		b[size] = a;
		delete[]mas;
		size = size + 1;
		mas = new int[size];
		for (int i = 0; i < size; i++)
		{
```

```
                        mas[i] = b[i];
                }
                delete[]b;
        }
        ~Massive()
        {
                delete[]mas;
                cout << "Massive was deleted"<<endl;
        }
};

class Stack : public Massive
{
public:
        Stack(int a) : Massive(a) {}
};

class Queque : public Massive
{
public:
        Queque(int a) : Massive(a) {}
};

int main()
{
        Massive* hw[2]{};
        int a1 = 0, a2 = 0, b1 = 0, b2 = 0;
        cout << "Enter the stack's size: ";
        cin >> a1;
        cout << "Enter the queque's size ";
        cin >> a2;

        hw[0] = new Stack(a1);
        hw[1] = new Queque(a2);

        hw[0]->Print();
        hw[1]->Print();

        cout << "Enter what element you want to add to the stack: ";
        cin >> b1;
        cout << "Enter what element you want to add to the queque: ";
        cin >> b2;

        hw[0]->AddElement(b1);
        hw[1]->AddElement(b2);
        hw[0]->Print();
        hw[1]->Print();

        delete hw[0];
        delete hw[1];
        return 0;
}
```

Варианты

1. Разработать программу с использованием наследования классов, реализующую классы: графический объект; круг; квадрат. Используя

виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран его размер и координаты.

2. Разработать программу с использованием наследования классов, реализующую классы: железнодорожный вагон; вагон для перевозки автомобилей; цистерна. Используя виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран его вес и количество единиц товара в вагоне.

3. Разработать программу с использованием наследования классов, реализующую классы: растения; культурные растения; сорняки. Используя виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран количество элементов и добавьте элемент.

4. Разработать программу с использованием наследования классов, реализующую классы: воин; пехотинец(винтовка); матрос(кортик). Используя виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран его возраст и вид оружия.

5. Разработать программу с использованием наследования классов, реализующую классы: точка; линия; круг. Используя виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран координаты и размер.

6. Разработать программу с использованием наследования классов, реализующую классы: работник больницы; медсестра; хирург. Используя виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран возраст и название должности.

7. Разработать программу с использованием наследования классов, реализующую классы: точка; квадрат, пирамида. Используя виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран его размер и координаты.

8. Разработать программу с использованием наследования классов, реализующую классы: реагент; углерод; железо. Используя виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран его количество и свойства (форма кристаллической решетки для углерода и чистота выработки руды для железа).

9. Разработать программу с использованием наследования классов, реализующую классы: работник фирмы; стажер; руководящий сотрудник; директор. Используя виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран целое число - уровень допуска, и название должности.

10. Разработать программу с использованием наследования классов, реализующую классы: молодой человек; студент; военнослужащий; военный курсант. Используя виртуальное наследование и виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран сведения о военнообязанности.

11. Разработать программу с использованием наследования классов, реализующую классы: периодические научные издания; журналы; материалы научных конференций. Используя виртуальное наследование и виртуальные

функции, не зная с объектом какого класса вы работаете, выведите на экран сведения о названии научного издания и количестве страниц.

12. Разработать программу с использованием наследования классов, реализующую классы: принтер; струйный принтер; лазерный принтер. Используя виртуальное наследование и виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран сведения о производителе и стоимости .

13. Разработать программу с использованием наследования классов, реализующую классы: транспортное средство; автомобиль; автобус. Используя виртуальное наследование и виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран сведения о пробеге и дате выпуска.

14. Разработать программу с использованием наследования классов, реализующую классы: точка, окружность, шар. Используя виртуальное наследование и виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран его размер и координаты.

15. Разработать программу с использованием наследования классов, реализующую классы: персональный компьютер, ноутбук, планшет. Используя виртуальное наследование и виртуальные функции, не зная с объектом какого класса вы работаете, выведите на экран его размер оперативной памяти и тактовую частоту процессора.

Полезные ссылки:

1. Уроки по C++. Базовое наследование. Режим доступа: https://ravesli.com/urok-154-bazovoe-nasledovanie-v-c/,

2. C++ для начинающих, глава 17. Режим доступа: https://cpp.com.ru/lippman/c17.html

3. Программирование на C++. Наследование и виртуальные функции. Режим доступа: https://bdpx.github.io/cpp/lab2_virtfunc.html

2.4. Перегрузка операций

Для всех вариантов необходимо редактировать классы, созданные в лабораторной работе №10.

Пример реализации

```
/*Переопределить операции << , >>, +, -, *  для ввода и вывода, 
сложения, вычитания и скалярного произведения объектов-векторов в 
двумерном пространстве.*/
# include <iostream>
# include <math.h>
using namespace std;

class AB
{
```

```
private:
    int x;
    int y;

public:
    int d;
    AB(int a,int b)
    {
        x=a;
        y=b;
    }
    AB(int a)
    {
        d=a;
    }
    AB()
    {
        x = 0;
        y = 0;
    }
    ~AB(){}
    AB operator+ (AB q)
      {
        int a = x + q.x;
        int b = y + q.y;
             return AB(a,b);
      }
      AB operator- (AB q)
      {
        int a = x - q.x;
        int b = y - q.y;
             return AB(a,b);
      }
      const AB operator* (const AB& q)
      {
        int d;
        d = x * q.x + y * q.y ;
             return AB(d);
      }
      friend ostream& operator<<(ostream& out, AB& q)
      {
        out << "x = " << q.x << endl;
        out << "y = " << q.y << endl;
             return (out);
      }
      friend istream& operator>> (istream& in, AB& q)
      {
        cout <<endl<< "x = ";
        in>>q.x;
        cout <<endl;
             return (in);
      }
};
```

```
int main()
{
    setlocale(LC_ALL, "rus");
    AB first, second;
    cout << "Ввод координат первого вектора"<<endl;
    cin>>first;
    cout << "Координаты первого вектора"<<endl<<first<<endl;
    cout << "Ввод координат второго вектора"<<endl;
    cin>>second;
    cout << "Координаты вектора вектора"<<endl<<second<<endl;
    AB slzh=first+second;
    AB rznst=first-second;
    AB pr=first*second;
    cout<<"координаты вектора сложения двух
векторов"<<endl<<slzh<<endl;
    cout<<"координаты вектора разности двух
векторов"<<endl<<rznst<<endl;
    cout<<"скалярное произведение равно: "<<pr.d<<endl;
    return 0;
}
```

Варианты

1. Переопределить операции << и >> для ввода-вывода объектов класса "Почтовый адрес", а также а также операцию == для проверки соответствия двух объектов(совпадают все данные: индекс, регион, населенный пункт, улица, номер дома и квартиры).

2. Переопределить операции <<, >>, +, -, * для ввода-вывода, сложения, вычитания и произведения объектов класса complex (класс для представления комплексных чисел).

3. Переопределить операции << , >>, +, -, * для ввода и вывода, сложения, вычитания и скалярного произведения объектов-векторов в трёхмерном пространстве.

4. Переопределить операции <<,>>, * для ввода, вывода и пересечения объектов класса "Прямоугольник".

5. Переопределить операции << , >>, +, -, * для ввода и вывода, сложения, вычитания и умножения объектов класса "Многочлен".

6. Переопределить операции << ,>>, +, -, * для ввода-вывода, сложения, вычитания и умножения матриц размерностью m*n, где m и n константы.

7. Переопределить операции <<, >>, +, - для ввода, вывода, сложения и вычитания объектов класса data. Результатом операции вычитания может выступать количество дней (часов, минут, секунд и др.) с обязательным указанием единиц измерения.

8. Переопределить операции << ,>>, +, -, [] для ввода-вывода, сложения, вычитания и доступа к элементу по индексу для объектов - одномерных целочисленных массивов.

9. Переопределить операции << ,>>, +, -, [] для ввода, вывода, конкатенации, вычитания и доступа к элементу по индексу для объектов - строк.

10. Переопределить операции << ,>>, +, -, * для ввода, вывода, сложения, вычитания и умножения объектов класса "Обыкновенная дробь".

11. Переопределить операции << и >> для ввода и вывода объектов класса "Студент", а также операцию == для проверки соответствия двух объектов (совпадают все персональные данные).

12. Объект «колода карт» представляет собой массив структур, где первое поле — масть, второе — значимость. Используя операции << и >> разработать механизм сохранения состояния колоды в файле и восстановления из файла. Перегрузить операцию == для проверки соответствия двух колод (совпадает порядок следования карт).

13. Структура «абонент телефонной сети» содержит следующие поля: 1) фамилия (строка), 2) улица, 3) дом, 4) квартира, 5) тел. номер. Переопределить операции << и >> для файлового ввода-вывода такого типа данных, а также операцию == для проверки соответствия двух объектов(совпадают фамилия, улица, дом, квартира и телефонный номер).

14. Переопределить операции << и >>, == для файлового ввода-вывода объектов типа «склад», где хранятся объекты типа «товар», а также операции проверки идентичности двух объектов "товар" (совпадают артикулы, количество, цена).

15. Переопределить операции << и >> для ввода и вывода объектов класса "Поздравительная открытка", а также операции == для проверки соответствия двух объектов(совпадают адресат, событие, текст и дата).

Полезные ссылки

1. Язык C++. Перегрузка операций. Режим доступа: https://prog-cpp.ru/cpp-operator/

2.5. Шаблоны классов

Пример реализации

```
/*
Написать программу, в которой описана иерархия классов: человек
(«дошкольник», «школьник», «студент», «работающий»).
Описать шаблонный класс для хранения массива указателей на объекты
произвольного класса, в шаблонном классе перегрузить операцию «[
]».
Продемонстрировать работу операторов и использование шаблонного
класса с различными классами.
*/

#include <iostream>

using namespace std;

class Humans
{
public:
```

```
	virtual void Print() {};
};

class Preschooler : public Humans
{
private:
	string hum;
public:
	Preschooler()
	{
		hum = "Preschooler";
	}
	void Print()
	{
		cout << "Human: " << hum << endl;
	}
};

class Schoolboy : public Humans
{
private:
	string hum;
public:
	Schoolboy()
	{
		hum = "Schoolboy";
	}
	void Print()
	{
		cout << "Human: " << hum << endl;
	}
};

class Student : public Humans
{
private:
	string hum;
public:
	Student()
	{
		hum = "Student";
	}
	void Print()
	{
		cout << "Human: " << hum << endl;
	}
};

class Worker : public Humans
{
private:
	string hum;
public:
```

```
	Worker()
	{
		hum = "Worker";
	}
	void Print()
	{
		cout <<"Human: "<< hum << endl;
	}
};

template <class T>
class Pointers
{
private:
	int length;
public:
	T* array;
	Pointers(int count)
	{
		length = count;
		array = new T[length];
	}

	T* operator[] (int n)
	{
		return array[n];
	}

	~Pointers()
	{
		delete[] array;
		array = nullptr;
	}
};

int main()
{
	Preschooler Obj1;
	Schoolboy Obj2;
	Student Obj3;
	Worker Obj4;

	Pointers <Humans*> l(4);

	l.array[0] = &Obj1;
	l.array[0]->Print();

	l.array[1] = &Obj2;
	l.array[1]->Print();

	l.array[2] = &Obj3;
	l.array[2]->Print();
```

```
        l.array[3] = &Obj4;
        l.array[3]->Print();

        return 0;
}
```

Варианты

1. Написать программу, в которой описана иерархия классов: ошибка в программе («ошибка доступа к памяти», «математическая», «деление на ноль», «переполнение»). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.
2. Написать программу, в которой описана иерархия классов: средство передвижения (велосипед, автомобиль, грузовик). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.
3. Написать программу, в которой описана иерархия классов: ошибка в программе («недостаточно памяти», «ошибка ввода/вывода», «ошибка чтения файла», «ошибка записи файла»). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.
4. Написать программу, в которой описана иерархия классов: геометрические фигуры (точка, прямая, треугольник, прямоугольник). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами
5. Написать программу, в которой описана иерархия классов: геометрические фигуры (точка, эллипс, четырехугольник). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.
6. Написать программу, в которой описана иерархия классов: геометрические фигуры (куб, цилиндр, тетраэдр). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.
7. Написать программу, в которой описана иерархия классов: геометрические фигуры (конус, шар, пирамида). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.

8. Написать программу, в которой описана иерархия классов: числа (целое, вещественное, комплексное). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.

9. Написать программу, в которой описана иерархия классов: треугольник (равнобедренный, равносторонний, прямоугольный). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.

10. Написать программу, в которой описана иерархия классов: прогрессия (арифметическая, геометрическая). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса различными классами.

11. Написать программу, в которой описана иерархия классов: геометрические фигуры (круг, параллелепипед, пирамида). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.

12. Написать программу, в которой описана иерархия классов: геометрические фигуры (ромб, прямоугольник, эллипс). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.

13. Написать программу, в которой описана иерархия классов: функция от одной переменной (синус, косинус, тангенс). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.

14. Написать программу, в которой описана иерархия классов: функция от одной переменной (логарифм, натуральный логарифм, а также класс, необходимый для представления производных). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.

15. Написать программу, в которой описана иерархия классов: функция от одной переменной (экспонента, гиперболический синус, гиперболический косинус). Описать шаблонный класс для хранения массива указателей на объекты произвольного класса, в шаблонном классе перегрузить операцию «[ ]». Продемонстрировать работу операторов и использование шаблонного класса с различными классами.

Полезные ссылки

1. Мясников Е.В., Попов А.Б. Шаблоны в C++: электронные методические указания к лабораторной работе. Самара, 2011. 24 с. Режим доступа: http://repo.ssau.ru/bitstream/Metodicheskie-ukazaniya/Shablony-v-C-Elektronnyi-resurs-elektron-metod-ukazaniya-k-lab-rabote-5-53577/1/%D0%9C%D1%8F%D1%81%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2%20%D0%95.%D0%92.%20%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B%20%D0%B2%20C%2B%2B.pdf

2. Шаблоны. Шаблоны класса. Режим доступа https://metanit.com/cpp/tutorial/9.1.php

2.6. Обработка исключительных ситуаций

В ходе выполнения задания необходимо:

1) реализовать класс, перегрузить для него операции, указанные в варианте;

2) определить исключительные ситуации;

3) предусмотреть генерацию исключительных ситуаций

Исключительные ситуации должны генерироваться:

1) в конструкторе с параметром при попытке создать вектор больше максимального размера;

2) в операции [] – при попытке обратиться к элементу с номером меньше 0 или больше текущего размера вектора(обработать 2 ситуации);

3) в операции + при попытке добавить элемент с номером больше максимального размера;

4) в операции – при попытке удалить элемент из пустого вектора.

Пример реализации

```
/*Класс- контейнер ВЕКТОР с элементами типа int. Реализовать операции:[] доступа по индексу;+ + добавляет элемент в вектор (постфиксная операция добавляет элемент в конец, префиксная в начало).*/
#define MAX_SIZE 50

#include <iostream>
#include <string>

using namespace std;

class Exception
{
private:
    string error;

public:
    Exception(string Error)
    {
        error = Error;
    }
```

```
    const char* getError()
    {
        return error.c_str();
    }
};

class Vector
{
private:
    int* array = { 0 };
    int size = 0;
public:
    Vector()
    {
        size = 1;
        array = new int[size];
    }
    Vector(int Size)
    {
        if (Size <= MAX_SIZE && Size > 0)
        {
            size = Size;
            array = new int[size];
            for (int i = 0; i < size; i++)
            {
                array[i] = i;
            }
        }
        else
        {
            throw Exception("Incorrect size");
        }
    }
    int operator[] (const int n)
    {
        if (n >= 0 && n < size)
        {
            return array[n];
        }
        else
        {
            throw Exception("Incorrect index");
        }
    }
    friend void operator ++ (Vector& vector)
    {
        if (1 + vector.size < MAX_SIZE)
        {
            int* array_new = new int[vector.size + 1];
            for (int i = 0; i < vector.size; i++)
            {
                array_new[i + 1] = vector.array[i];
            }
```

```
            array_new[0] = 0;
            delete[] vector.array;
            vector.array = array_new;
            vector.size += 1;
        }
        else
        {
            throw Exception("Incorrect new size");
        }
    }

    friend void operator ++ (Vector& vector, int)
    {
        if (1 + vector.size < MAX_SIZE)
        {
            int* array_new = new int[vector.size + 1];
            for (int i = 0; i < vector.size; i++)
            {
                array_new[i] = vector.array[i];
            }
            array_new[vector.size] = 0;
            delete[] vector.array;
            vector.array = array_new;
            vector.size += 1;
        }
        else
        {
            throw Exception("Incorrect new size");
        }
    }

    void print()
    {
        for (int i = 0; i < size; i++)
        {
            cout << array[i] << " | ";
        }
        cout << endl;
    }
    int getSize()
    {
        return size;
    }
    ~Vector()
    {
        delete[] array;
    }

};

int main()
{
```

```
    try
    {
        Vector array(10);
        array.print();
        array++;
        ++array;
        array.print();
    }
    catch (Exception& exception)
    {
        cerr << "Error: " << exception.getError() << endl;
    }
    return 0;
}
```

Варианты

1. Класс- контейнер ВЕКТОР с элементами типа int. Реализовать операции:[]– доступа по индексу; () – определение размера вектора;+ число – добавляет константу ко всем элементам вектора;-n-удаляет n элементов из конца вектора.
2. Класс- контейнер ВЕКТОР с элементами типа int. Реализовать операции:[]– доступа по индексу; () – определение размера вектора; *число – умножает на константу все элементы вектора; -- - удаляет последний элемент из вектора.
3. Класс- контейнер ВЕКТОР с элементами типа int. Реализовать операции:[]– доступа по индексу; int() – определение размера вектора;-n – удаляет n элементов из конца вектора;+n-добавляет n элементов в конец вектора.
4. Класс- контейнер ВЕКТОР с элементами типа int.Реализовать операции:[] – доступа по индексу;() – определение размера вектора;-- удаляет элемент из вектора (постфиксная операция удаляет элемент в конце вектора, префиксная – в начале).
5. Класс- контейнер ВЕКТОР с элементами типа int.Реализовать операции:[] – доступа по индексу;int() – определение размера вектора;* вектор –умножение элементов векторов a[i]*b[i];+n– переход вправо к элементу с номером n
6. Класс- контейнер МНОЖЕСТВО с элементами типа int.Реализовать операции:[] – доступа по индексу;() – определение размера множества;+ – объединение множеств;++ - добавление элемента в множество.
7. Класс- контейнер МНОЖЕСТВО с элементами типа int.Реализовать операции:[] – доступа по индексу;int() – определение размера вектора;*– пересечение множеств;--- удаление элемента из множества.
8. Класс- контейнер МНОЖЕСТВО с элементами типа int. Реализовать операции:[] – доступа по индексу;== - проверка на равенство;> число – принадлежность числа множеству;- n - переход влево к элементу с номером n

9. Класс- контейнер МНОЖЕСТВО с элементами типа int. Реализовать операции:[] – доступа по индексу;!= - проверка на неравенство;< число – принадлежность числа множеству;+n– переход вправо к элементу с номером n .

10. Класс- контейнер МНОЖЕСТВО с элементами типа int. Реализовать операции:[] – доступа по индексу;() – определение размера вектора;-– разность множеств;---– удаление элемента из множества.

11. Класс- контейнер СПИСОК с ключевыми значениями типа int.Реализовать операции:[]– доступа по индексу;int() – определение размера списка;+ список –сложение элементов списков a[i]+b[i];-n- переход влево к элементу с номером n.

12. Класс- контейнер СПИСОК с ключевыми значениями типа int.Реализовать операции:[] – доступа по индексу;() – определение размера списка;+ число – добавляет константу ко всем элементам списка;++ - добавление элемента в конец списка.

13. Класс- контейнер СПИСОК с ключевыми значениями типа int.Реализовать операции:[]– доступа по индексу;+ вектор– добавление списка b к списку a (a+b)+ число – добавляет элемент в начало списка.

14. Класс- контейнер СПИСОК с ключевыми значениями типа int. Реализовать операции:[] – доступа по индексу;() – определение размера списка;* число – умножает все элементы списка на число;- n – переход влево к элементу с номером n.

15. Класс- контейнер СПИСОК с ключевыми значениями типа int.Реализовать операции:[] – доступа по индексу;int() – определение размера списка;* список –умножение элементов списков a[i]*b[i];+n - переход вправо к элементу с номером n.

Полезные ссылки

1. Уроки по C++. Исключения. Зачем они нужны? Режим доступа: https://ravesli.com/urok-181-isklyucheniya-zachem-oni-nuzhny/

2. Уроки по C++. Обработка исключений. Операторы throw, try и catch. Режим доступа: https://ravesli.com/urok-182-obrabotka-isklyuchenij/

3. Уроки по C++. Исключения, функции и раскручивание стека. Режим доступа: https://ravesli.com/urok-183-isklyucheniya-funktsii-i-raskruchivanie-steka/

ЗАКЛЮЧЕНИЕ

Программирование – настолько обширная область, что охватить все нюансы в данном практикуме не представляется возможным. Автор полагает, что материалы пособия помогут начинающим программистам сориентироваться в огромном информационном пространстве, посвященном искусству разработки программ.

В практикуме рассматриваются самые простые вопросы, азы программирования, при этом все задания в главах разнесены по уровням сложности; в рамках одного задания уровень сложности одинаковый.

В практикуме не затронута разработка многофайловых проектов, использования средств отладки и контроля данных – всё это, безусловно, очень важные темы, которые необходимо знать профессиональным программистам.

И, в заключении, хотелось бы принести благодарность своим пытливым, упорным и замечательным студентам, оказавшим помощь в подготовке данного практикума: Алексею Серёгину, Дмитрию Гильману, Егору Дмитриенко, Никите Юскову, Никите Кураеву, Алексею Горбунову, Ильясу Жунусову, Андрею Гмызину, Егору Голикову, Арсению Смольникову.

ГЛОССАРИЙ

Абстрагирование - процесс выделения абстракций в предметной области задачи.

Абстрактный класс - базовый класс, который не предполагает создания экземпляров

Абстрактный метод - метод класса, который объявляется, но не реализуется в классе.

Абстракция – совокупность существенных характеристик некоторого объекта, которые отличают его от всех других видов объектов и, таким образом, четко определяют особенности данного объекта с точки зрения дальнейшего рассмотрения и анализа.

Алгоритм – точное и конечное описание того или иного общего метода, основанного на применении исполнимых элементарных тактов обработки.

Алгоритм Дейкстры – служит для поиска кратчайшего пути в сети.

Аппарат защиты – прогнозирование поведения объектов и контроль за соблюдением ограничений.

Аппарат развития – определение и использование новых абстракций или уточнение и конкретизация уже имеющихся абстракций.

Атомарный элемент – элемент структуры, представляющий собой отдельную запись об объекте, является неделимым.

Базис – встроенные в язык программирования знаки и их денотаты. В базис входят элементарные операции, структурные типы данных, структурные операции.

Библиотечные функции - функции, описанные в библиотеках языка С (как стандартных, так и не стандартных).

Бинарное дерево – упорядоченное дерево, в котором задан порядок следования узлов и каждый узел имеет не более двух потомков.

Бинарный Поиск – аргумент поиска сравнивается с элементом, расположенным в медиане, при их совпадении фиксируется удачный поиск, при не совпадении принимается решение о том, в какой части массива продолжать поиск. Если аргумент поиска меньше, чем элемент в медиане, то поиск продолжается слева от медианы, а если больше – справа от медианы. Неудачный поиск фиксируется, когда левая граница становится больше, чем правая.

В – дерево – порядка n представляет собой совокупность иерархически связанных страниц внешней памяти (каждая вершина дерева - страница), обладающая следующими свойствами: каждая страница содержит не более 2*n элементов (записей с ключом); каждая страница, кроме корневой, содержит не менее n элементов; если внутренняя (не листовая) вершина B-дерева содержит m ключей, то у нее имеется m+1 страниц-потомков; все листовые страницы находятся на одном уровне.

Ветвление – алгоритмическая структура, в которой осуществляется выбор одного из двух возможных действий в зависимости от условия. Если условие

истинно, то выполняется одна группа команд, в противном случае – другая. Используются как полная, так и сокращенная форма команды ветвления.

Взаимная рекурсия – если процедура a рекурсивно вызывает процедуры a и b, процедура b в свою очередь рекурсивно вызывает a и b, то говорят о взаимной рекурсии.

Виртуальный метод - метод (функция) класса , который может быть переопределён в классах-наследниках так, что конкретная реализация метода для вызова будет определяться во время исполнения**;** метод базового класса, который могут переопределить классы-наследники так, что конкретный адрес вызываемого метода определяется только при выполнении программы.

Восходящее программирование - процесс разработки программ, при котором сложная программа реализуется посредством интеграции более простых подпрограмм, начиная с элементарных подпрограмм

Временная эффективность – программы по соответствующему алгоритму определяется временем, необходимым для ее выполнения.

Глобальные переменные – переменные, объявленные вне какой-либо функции. В отличие от локальных переменных глобальные переменные могут быть использованы в любом месте программы, но перед их первым использованием они должны быть объявлены. Область действия глобальной переменной – вся программа.

Двунаправленный список – список, в котором каждый элемент имеет два поля связи с предыдущим и следующим элементом.

Декларативное программирование - парадигма программирования, в которой задаётся спецификация решения задачи, то есть описывается, что представляет собой проблема и ожидаемый результат.

Денотат – это значение или обозначение.

Дерево – представляет собой иерархическую структуру, состоящую из узлов с наложенными на них отношениями подчиненности. Каждый узел дерева может иметь несколько потомков, являясь для них предком.

Деструктор - метод класса, который предназначен для уничтожения экземпляров класса, а также для освобождения ресурсов используемых в объектах класса (например, освобождение памяти).

Детерминированность алгоритма – система величин, получаемых в какой-то момент времени не начальный, однозначно определяется системой величин полученных в предшествующие моменты времени, т.е. каждый последующий шаг полностью определяется предыдущим. Другими словами, детерминированность алгоритма является процессом строго направленным, полностью управляемым, не допускающим произвола.

Динамическая память – участок ОЗУ ПК, который может использоваться программой для обработки больших массивов данных.

Динамическая переменная – переменная, созданием и уничтожением которой может явно управлять программист.

Динамические структуры данных – используются для хранения и обработки данных, интенсивно изменяющихся по количеству элементов.

Дискретность алгоритма – это процесс последовательного построения величин, идущий в дискретном времени таким образом, что в начальный момент задается исходная конечная система величин, а в каждый следующий момент система величин, получается, по определенному закону из системы величин, имевшихся в предыдущий момент времени.

Дружественная функция - функция, которая имеет доступ к закрытым членам класса, как если бы она сама была членом этого класса.

Дружественность – простота установки програмного средства, интуитивно понятный пользовательский интерфейс, детальная контекстная справочная система, наличие вспомогательных сервисных функций (масштабирование рисунков, вывод калькулятора, возможность возврата к предыдущим вопросам и т.д.).

Дуга – упорядоченная пара вершин.

Естественное слияние – метод сортировки, который основывается на распознавании серий при распределении и их использовании при последующем слиянии. Серией называется подпоследовательность записей ai, a(i+1), ..., aj такая, что ak <= a(k+1) для всех i <= k < j, ai < a(i-1) и aj > a(j+1).

Заголовок модуля - интерфейсная часть, представленная в виде файла с расширением .h.

Заголовок функции - описание интерфейсной части функции, которая содержит: тип возвращаемого значения, имя функции и список формальных параметров функции.

Закольцованная очередь – при достижении одним из указателей конца зарезервированного блока этот указатель перебрасывается на первую ячейку.

Запись – структурированный тип данных, состоящий из фиксированного числа компонентов различных типов.

Знак – называют именем или обозначением, денотат это значение или обозначение.

Знаковая система – правила образования знаков (синтаксис) и согласованные с ними правила образования денотатов (семантика).

Иерархичность – принцип ООП, заключающийся в ранжировании или упорядочении системы абстракций. Принцип иерархичности предполагает использование иерархий при разработке программных систем.

Изолированная вершина графа – вершина, у которой нет ни одной смежной вершины.

Имя переменной – любая последовательность прописных и строчных букв английского алфавита и символа подчеркивания '_'. По умолчанию значимыми являются первые 32 символа.

Инкапсуляция – сочетание объединения всех свойств предмета (составляющих его состояния и поведения) в единую абстракцию, а также ограничения доступа к реализации этих свойств; скрытие внутреннего устройства объектов.

Интерполяционный поиск – метод поиска исходит из того, что интервал поиска делится не пополам. В общем случае предполагаемое место нахождения

аргумента поиска вычисляется исходя из предположения, что ключи в массиве упорядочены и имеют постоянный шаг.

Интерпретатор (англ. interpreter — истолкователь) – переводит и выполняет программу строка за строкой.

Интерфейс – описание того, как можно взаимодействовать с окружающим миром.

Интерфейс функции - заголовок функции, в котором указывается название функции, список ее параметров и тип возвращаемого значения

Интерфейсная часть модуля – содержит объявление глобальных объектов модуля (констант, типов, переменных, процедур и функций), которые станут доступными, как в основной программе, так и в других модулях.

Итерация – многократное исполнение одних и тех же действий.

Каскадная рекурсия – если, по меньшей мере, на одной ветке разветвления в теле рекурсии встречаются два и более рекурсивных вызова, то говорят о каскадной рекурсии.

Качество программных средств – совокупность его черт и характеристик, которые влияют на его способность удовлетворять заданные потребности пользователей.

Класс – производный тип, который задает некоторую совокупность данных и позволяет определить набор операций над этими данными; множество объектов, имеющих общую структуру и поведение; это сущность, которая задает общие свойства и общее поведение для объектов (общий шаблон для создания объектов).

Ключ поиска – уникальный ключ, который однозначно идентифицирует элемент.

Компилятор (англ. compiler – составитель, собиратель) – читает всю программу целиком, делает ее перевод и создает законченный вариант программы на машинном языке, который затем и выполняется.

Конструктор - метод класса, который всегда вызывается при создании экземпляра класса (объекта)

Конструктор копирования - специальный конструктор, применяемый для создания нового объекта как копии уже существующего.

Корень – узел в дереве не имеющий предка.

Линейная рекурсия – каждый рекурсивный вызов приводит не более, чем к одному дальнейшему рекурсивному вызову.

Линейность – свойство структур данных означающее, что элементы в структуре не зависят друг от друга.

Линейный поиск – аргумент поиска последовательно сравнивается с ключами, при совпадении фиксируется удачный поиск и определяется номер искомого ключа. После полного перебора элементов фиксируется неудачный поиск.

Лист – узел дерева, не имеющий потомков.

Литера – один символ, заключенный в апострофы.

Логическое представление – описание отдельных составляющих исходной структуры представляющих собой, более простые структуры, а также

представление операций исходной структуры через операции над составляющими ее простыми структурами.

Локальная переменная – переменная, объявленная внутри подпрограммы.

Максимально-ориентированный лес – лес, в котором учтена ориентация, т.е. от какого ребра пойдем.

Массив - пронумерованная последовательность величин одинакового типа, обозначаемая одним именем; конечный (фиксированный), упорядоченный набор однородных элементов с произвольным доступом

Массовость алгоритма – свойство алгоритма, означающее, что начальная система величин может выбираться из некоторого потенциально бесконечного множества.

Метаязык – надязыки используемые для описания синтаксиса языка.

Методология программирования – совокупность механизмов, применяемых в процессе разработки программного обеспечения и объединенных одним общим философским подходом.

Метод - - процедура или функция, принадлежащая классу объектов.

Методология структурного программирования – в основе лежит последовательная декомпозиция (разложение) исходной задачи на ряд подзадач. Основной идеей структурного программирования является ориентация в первую очередь на описание процесса.

Методы проектирования – выделяют три основных метода проектирования: нисходящее, восходящее и метод пошаговой детализации.

Модуль - совокупность типов данных, переменных, констант и функций для работы с этими типами данных.

Многопутевое слияние – метод внешней сортировки является распределение серий исходного файла по m вспомогательным файлам B1, B2, ..., Bm и их слияние в m вспомогательных файлов C1, C2, ..., Cm. На следующем шаге производится слияние файлов C1, C2, ..., Cm в файлы B1, B2, ..., Bm и т.д., пока в B1 или C1 не образуется одна серия.

Множество – наборы однотипных объектов, каким-либо образом связанных между собой.

Мобильность – способность ПС быть перенесенным из одной среды (окружения) в другую, в частности, с одного компьютера на другой.

Модуль – набор связанных функций вместе с данными, которые они обрабатывают.

Морфология – опознает формы элементов языка.

Мощность алфавита – общее количество символов в алфавите (N).

Наследование – такое отношение между классами, когда один класс повторяет структуру и поведение другого класса (одиночное наследование) или других (множественное наследование) классов.

Надежность – способность ПС выполнять заданные функции в соответствии с программными документами в условиях, вызванных сбоями технических средств, ошибками во входных данных, ошибками обслуживания и другими дестабилизирующими воздействиями

Неориентированный граф – граф, содержащий только ребра.

Нетипизированный файл – содержит записи различной длины, либо не имеющий определенной структуры, в этом случае файл можно считывать и записывать блоками. Отсутствие типа делает эти файлы совместимыми с любыми другими типами файлов и позволяет организовать высокоскоростной обмен данными.

Нисходящее программирование - процесс разработки программ, при котором сложная программа разбивается на ряд более простых подпрограмм, которые в свою очередь также могут быть разбиты на ряд еще более простых подпрограмм.

Область действия переменной – правила, которые устанавливают, какие данные доступны из определенного места программы.

Объединение - переменная, которая может содержать (в разные моменты времени) объекты различных типов и размеров.

Объект – особый опознаваемый предмет, блок или сущность (реальная или абстрактная), имеющая важное функциональное назначение в данной предметной области; сущность в адресном пространстве вычислительной системы, которая появляется при создании экземпляра класса и обладает определенным состоянием, уникальностью и поведением.

Объектно–ориентированная методология – определяется как технология создания сложного программного обеспечения, которая основана на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного типа (класса), а классы образуют иерархию с наследованием свойств.

Объектно-ориентированное программирование - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Ограничение доступа – принципы ООП, заключающийся в сокрытии отдельных элементов реализации абстракции, не затрагивающих существенных характеристик ее как целого.

Однонаправленный список – список, в котором каждый элемент имеет только одно поле связи с предыдущим или следующим элементом.

Однородное дерево – дерево, каждый узел которого может быть представлен одним и тем же типом записи.

Однородный – свойство структур данных означает, что все элементы структуры принадлежат одному и тому же типу данных.

Оператор – определяется как некоторый логически законченный самостоятельный этап процесса обработки данных.

Оператор присоединения – в Паскале служит для приписывания единожды указанного имени (до точки) последующим именам полей.

Оператор цикла – служит для обозначения повторяющихся фрагментов алгоритма, используют специальную конструкцию, называемую циклом.

Различают три вида операторов цикла: оператор for, оператор while, оператор repeat … until.

Операция разыменования – конструкция, которая служит для реализации доступа к переменной через указатель на нее. Необходимо ПОСЛЕ переменной-указателя поставить знак “^”. Запись P^ – означает «переменная, на которую ссылается P». Разыменование допускается для любых ссылочных типов.

Ориентированное дерево – граф не содержащий циклов, в котором только одна вершина имеет нулевую степень захода (в неё не ведут дуги), а все остальные вершины имеют степень захода 1 (в них ведёт ровно по одной дуге).

Ориентированный граф – граф, содержащий только дуги.

Отладка – заключается в определении мест возникновения ошибок, выяснении причины возникновения ошибок и их устранении. Эта деятельность носит негативный характер в том смысле, что она полностью направлена на избавление от обнаруженных ошибок.

Очередь – линейная, однородная структура переменного размера, элементы которой удаляются с одного конца, называемого началом очереди, и помещаются в другой конец, называемый концом очереди.

Парадигма - совокупность ценностей, методов, подходов, технических навыков и средств, принятых в научном сообществе в рамках устоявшейся научной традиции в определенный период времени

Параллелизм – свойство нескольких абстракций одновременно находиться в активном состоянии, т.е. выполнять некоторые операции.

Парадигма программирования - совокупность идей и понятий, определяющих стиль написания программ (подход к программированию).

Параметры функции - значения, передаваемые в функцию при ее вызове

Пересечение – операция, определенная над множественным типом, которая возвращает в качестве результата только общие для заданных операндов элементы.

Перечисляемый тип – задается перечислением тех значений, которые он может получать. Каждое значение именуется идентификатором и располагается в списке, заключенном в круглые скобки. Максимальный размер перечисляемого типа составляет 65 536 значений.

Повторная рекурсия – перед новым выполнением рекурсивного вызова, вычисление предыдущего рекурсивного вызова прекращается.

Поиск в последовательности – процесс нахождения элемента в последовательности по значению одного или более чем одного поля.

Покрывающие дерево – является любое дерево, образованное совокупностью дуг, включающих все вершины графа.

Полиморфизм – свойство, которое позволяет одно и тоже имя использовать для решения нескольких технически разных задач.

Полиномиальный алгоритм – алгоритмы "эффективные", или "хорошие" алгоритмы, когда удается более глубоко проникнуть в суть решаемой задачи.

Полустепень захода узла – число ребер, для которых данная вершина является конечной.

Полустепень исхода узла – число ребер, исходящих из некоторой вершины.

Пользовательские функции - функции, реализованные программистом в процессе разработки программы.

Пользовательский интерфейс – представляет средство взаимодействия пользователя с ПС.

Порядковый тип – называют тип, для которого точно известно, какой элемент за каким следует.

Последовательное представление – структуры данных очередь моделируется в виде массива, в этом случае резервируется блок памяти, внутри которого, очередь может расти и сокращаться.

Постоянные –программные объекты, которые не изменяются в ходе выполнения программ.

Препроцессор - программа, осуществляющая обработку текста программы перед ее непосредственной компиляцией. Обработка осуществляется согласно специальным указаниям, называемым директивами препроцессора

Присваивание – операция, которая означает положить в ячейку памяти ПК.

Программа – последовательность инструкций (команд), выполняемых компьютером с целью получения результата.

Программный модуль – любой фрагмент описания процесса, оформляемый как самостоятельный программный продукт, пригодный для использования в описаниях процесса.

Пространственная характеристика (емкость) – измеряется количеством памяти, требуемой для выполнения программы соответствующего алгоритма.

Простые переменные – описываются структурами, состоящими из одного элемента, поэтому каждая простая переменная характеризуется одним значением. При отображении на память ПК имени простой переменной ставится в соответствие номер ячейки памяти, в которой хранится значение этой переменной.

Прототип функции - необязательная часть описания функции, предназначенная для объявления некоторой функции, интерфейс которой соответствует данному прототипу.

Прочность модуля – мера его внутренних связей.

Прямое слияние – сортировка состоит из последовательности шагов, в каждом из которых выполняется распределение состояния файла A в файлы B и C, а затем слияние файлов B и C в файл A. На первом шаге для распределения последовательно читается файл A, и записи a1, a3, ..., a(n-1) пишутся в файл B, а записи a2, a4, ..., an - в файл C (начальное распределение). Начальное слияние производится над парами (a1, a2), (a3, a4), ..., (a(n-1), an), и результат записывается в файл A. На втором шаге снова последовательно читается файл A, и в файл B записываются последовательные пары с нечетными номерами, а в файл C - с четными. При слиянии образуются и пишутся в файл A упорядоченные четверки записей. И так далее. После слияния файл A будет содержать полностью упорядоченную последовательность записей.

Размер модуля – измеряется числом содержащихся в нем строк или операторов.

Разность – операция, определенная над множественным типом, которая возвращает те элементы, которые входят в первый операнд, но не входят во второй. Например: d:=[2..5] - [4..6]. Результат: d= [2, 3].

Распределенное программирование – выделяют три уровня: распределенное программирование в малом; распределенное программирование в большом; распределенные прикладные программные системы. Распределенное программирование в малом связано с параллельным выполнением операторов одной программы (например, параллельным выполнением циклов). Распределенное программирование в большом, состоит в разработке распределенных программных систем. Распределенные прикладные программные системы предназначены для поддержки коллективных разработок и других видов коллективной деятельности в самых различных областях.

Ребро – неупорядоченная пара вершин.

Реализация функции - тело функции, содержащее внутренние (локальные) данные функции и программный код, выполняющий действия согласно переданным в функцию параметрам и возвращающий значение, соответствующего интерфейсу функции типа

Результативность алгоритма – исполнение алгоритма сводится к выполнению конечного числа действий и всегда приводит к решению задачи.

Рекурсивный объект – называется рекурсивным, если он содержит сам себя или определен с помощью самого себя.

Рекурсия - определение понятия через само это понятие.

Рутинность модуля – его независимость от предыстории обращения к нему.

Сбалансированное ДЕРЕВО – называют дерево, в котором каждый узел имеет одинаковое число ветвей.

Связанное представление – представление данных, обеспечивающее большую гибкость структуры, т.к. операции добавления и удаления данных выполняются без перезаписи массива. При связанном представлении дерево может расти неограниченно.

Семантика – устанавливает смысл предложения или строки.

Синтаксис – множество правил или формул, которые задают множество формально правильных предложений.

Синтаксическая диаграмма – графически изображает структуру тех конструкций, каждая из которых является значением определяемой метапеременной.

Следование – в данной структуре все команды выполняются одна за другой в том порядке, как они записаны, т.е. управление передается последовательно от одного процесса к другому.

Сложность алгоритма – оценка функции трудоемкости алгоритма, которая позволяет определить предпочтения в использовании того или иного алгоритма для больших значений размерности исходных данных.

Случайный поиск – алгоритм поиска, в котором предполагаемая точка нахождения ключа M назначается случайным образом.

Смешанный граф – граф, в котором есть ребра, и дуги.

Сопровождаемость – характеристики ПС, которые позволяют минимизировать усилия по внесению изменений для устранения в нем ошибок и по его модификации в соответствии с изменяющимися потребностями пользователей.

Сортировка – процесс перестановки объектов заданного массива в определенном порядке. Цель сортировки – облегчить последующий поиск элементов в отсортированном массиве.

Сортировка по возрастанию – расположение элементов от меньшего к большему.

Сортировка по убыванию – расположение элементов от большего к меньшему.

Сортировка слиянием – имеется несколько полностью заполненных и отсортированных массива. Требуется объединить элементы всех массивов в один массив, так чтобы элементы в нем были отсортированы. Количество исходных массивов определяет количество фаз в сортировке. Для каждого из исходных массивов назначается маркер, который отмечает претендента на включение в выходной массив. В выходной массив включается меньший по значению из двух претендентов, после чего маркер в этом массиве сдвигается. Если в процессе сортировки один из массивов заканчивается в выходной массив переписывается остаток другого массива.

Составной оператор – последовательность произвольных операторов программы, заключенная в операторные скобки.

Составной тип – последовательность произвольных операторов программы, заключенная в операторные скобки.

Спецификация – представляет собой полное и точное описание функций и ограничений разрабатываемого программного средства

Списковая структура – называется структура, представляющая собой список, элементами которого могут быть списки и списковые структуры.

Список магазинного типа – это линейный список, все элементы которого вставляются и удаляются только с одного или обоих концов.

Статические структуры данных – используются для хранения и обработки данных, не изменяющихся по количеству элементов (словари, справочники, справочные данные и т.д.).

Стек – линейная, однородная структура переменного размера, в котором все включения и исключения элементов производятся с одного конца - вершины стека.

Степень узла – число дуг, инцидентных узлу.

Строка – набор знаков, символов. Тип данных в языках программирования.

Структура - это одна или несколько переменных (в т.ч. различных типов), которые сгруппированы под одним именем.

Структура выбора – предназначена для выбора одного из многих вариантов. В данной структуре осуществляется сравнение ключа с одной из предложенных альтернатив, если совпадение произошло, выполняется соответствующая команда и происходит выход из структуры.

Структура данных – совокупность элементов информации, находящихся в определенной, заранее заданной взаимосвязи.

Структурная методология – написание программ в соответствии с заранее определенной дисциплиной.

Сцепление модуля – мера его зависимости по данным от других модулей.

Текстовый файл – совокупность строк различной длины.

Тело функции - часть-реализация, содержащая программный код, выполняемый при вызове функции.

Тестирование – процесс, посредством которого проверяется правильность программы. Эта деятельность носит позитивный характер, ее цель – показать, что программа правильно и действительно работает в соответствии с проектными спецификациями. Тестирование закончено, когда осуществлены все требуемые проверки на соответствие спецификации.

Тип данных – множество значений и множество операций над этими значениями.

Тип-диапазон – подмножество своего базового типа, в качестве которого может выступать любой порядковый тип, кроме типа-диапазона.

Типизация – ограничение, накладываемое на свойства объектов и препятствующее взаимозаменяемости абстракций различных типов (или сильно сужающее возможность такой замены).

Типизированный файл – файл прямого доступа, т.е. он может быть открыт и для записи и для чтения одновременно. Тип компонентов типизированного файла может быть любой тип, кроме файлового типа.

Транслятор (англ. translator – переводчик) – программа-переводчик, которая преобразует программу, написанную на одном из языков высокого уровня, в программу, состоящую из машинных команд.

Трудоёмкость алгоритма – для данного конкретного входа – (N), будем понимать количество «элементарных» операций совершаемых алгоритмом для решения конкретной проблемы в данной формальной системе.

Удаленная рекурсия – если в теле рекурсивного объявления в выражениях фактических параметров, в свою очередь, встречаются рекурсивные вызовы, то говорят об удаленной рекурсии.

Узел – вершина графа.

Указатель – переменная, предназначенная для хранения адреса в памяти (это просто целое число).

Указатель NULL – указатель, который «никуда не указывает», такая ссылка называется пустой ссылкой. Указатель NULL считается константой, совместимой с любым ссылочным типом.

Указатель на файловую переменную – указатель на информацию, которая определяет различные стороны файла: имя, статус, текущую позицию.

Упорядоченное дерево – дерево, в котором задан порядок следования узлов.

Упорядоченность – ключевое поле данных наследника слева меньше, чем ключевое поле предка, а ключевое поле правого наследника больше, чем предка.

Упорядоченный массив – массив, в котором все элементы упорядочены по номеру.

Устойчивость – свойство абстракции существовать во времени независимо от процесса, породившего данный программный объект, и/или в пространстве, перемещаясь из адресного пространства, в котором он был создан.

Файл – совокупность связанных записей, рассматриваемых как единое целое и хранящихся на внешнем носителе; именованная область данных на каком-либо носителе информации.

Фактические параметры - идентификаторы, указанные в строке вызова подпрограмм; переменные, выражения, константные значения или вызовы других функций, указываемые при непосредственном вызове функции внутри другой функции.

Физическое представление – определяет способы хранения структуры данных в памяти, а также способы доступа к элементам.

Фиксированный размер – свойство структур данных, которое означает что после создания структуры нельзя включать или исключать записи, а позволяют лишь их корректировать.

Формальные параметры – идентификаторы, веденные в заголовки подпрограмм; переменные, описываемые при объявлении функции в ее прототипе и заголовке и используемые в программном коде тела функции.

Функциональное программирование - метод программирования, основанный на разбиении алгоритма на отдельные функциональные модули и описания их связей и взаимодействия; метод построения программы как последовательности вложенных друг в друга вызовов функций

Функциональность – способность программного средства выполнять набор функций, удовлетворяющих заданным или подразумеваемым потребностям пользователей. Набор указанных функций определяется во внешнем описании программных средств.

Функция – самостоятельная единица программы, оформленная по правилам написания программ; синтаксически выделенный именованный программный модуль, выполняющий определенное действие или группу действий.

Хеширование – метод индексирования, согласно которому значение ключа подвергается числовым манипуляциям в целях непосредственного вычисления местоположения соответствующей записи в файле или исходной точки для поиска этой записи.

Часть реализации модуля – содержит тела подпрограмм, заголовки которых объявлены в интерфейсной части модуля, причем заголовки этих подпрограмм в исполнительной части (части реализации) могут быть записаны в сокращенном виде. В части реализации помещены рабочие объекты, называемые невидимыми или скрытыми.

Шаблон - это предписание для создания класса, в котором один или несколько типов либо значений параметризованы.

Экспоненциальный алгоритм – алгоритмы "неэффективные" просто варианты полного перебора.

Эффективность – отношение уровня услуг, предоставляемых ПС пользователю при заданных условиях, к объему используемых ресурсов.

Язык программирования – знаковая система для планирования поведения ПК.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Абрамян М.Э. 1000 задач по прораммированию / М.Э. Абрамян. В 3 ч. Ростов – на Дону, 2004 г. – Режим доступа: https://newlms.magtu.ru/mod/resource/view.php?id=921148
2. Быков А.А.Сборник задач по программированию с решениями. Режим доступа: https://newlms.magtu.ru/pluginfile.php/1575759/mod_resource/content/0/Sbornik_zadach_po_programmirovaniyu_s_resheniami.pdf
3. Варфоломеева Т.Н., Повитухин С.А. Задачник – практикум по программированию на примере языка СИ: [Электронный ресурс] : учебное пособие / Т. Н. Варфоломеева, С. А. Повитухин ; ФГБОУ ВО «Магнитогорский государственный технический университет им. Г.И. Носова». – Электрон. текстовые дан. (512 Кб). – Магнитогорск : ФГБОУ ВО «МГТУ», 2019.
4. Варфоломеева, Т. Н. Программирование : учебно-методическое пособие / Т. Н. Варфоломеева, С. А. Повитухин ; МГТУ. - Магнитогорск : МГТУ, 2015. - 1 электрон. опт. диск (CD-ROM). - Загл. с титул. экрана. - URL : https://magtu.informsystema.ru/uploader/fileUpload?name=3857.zip&show=dcatalogues/1/1123503/3857.zip&view=true (дата обращения: 08.12.2021). - Макрообъект. - Текст : электронный. - Сведения доступны также на CD-ROM.
5. Варфоломеева, Т. Н. Теоретические основы алгоритмизации программирования : учебное пособие / Т. Н. Варфоломеева, С. А. Повитухин ; МГТУ. - Магнитогорск : МГТУ, 2016. - 1 электрон. опт. диск (CD-ROM). - Загл. с титул. экрана. - URL : https://magtu.informsystema.ru/uploader/fileUpload?name=3858.zip&show=dcatalogues/1/1130374/3858.zip&view=true (дата обращения: 08.12.2021). - Макрообъект. - Текст : электронный. - Сведения доступны также на CD-ROM.
6. Воронцова, Е. А. Программирование на C++ с погружением: практические задания и примеры кода - Москва :НИЦ ИНФРА-М, 2016. - 80 с. ISBN 978-5-16-105159-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/563294 (дата обращения: 19.12.2021). – Режим доступа: по подписке.
7. Гниденко, И. Г. Технологии и методы программирования : учебное пособие для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва : Издательство Юрайт, 2021. — 235 с. — (Высшее образование). — ISBN 978-5-534-02816-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/469759 (дата обращения: 19.12.2021).
8. Златопольский Д. 1400 задач по программированию. – М.: ДМК Пресс, 2020. – 192 с.: ил
9. Зыков, С. В. Программирование : учебник и практикум для вузов / С. В. Зыков. — Москва : Издательство Юрайт, 2021. — 320 с. — (Высшее образование). — ISBN 978-5-534-02444-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/469579 (дата обращения: 19.12.2021).

10. Зыков, С. В. Программирование. Объектно-ориентированный подход : учебник и практикум для вузов / С. В. Зыков. — Москва : Издательство Юрайт, 2021. — 155 с. — (Высшее образование). — ISBN 978-5-534-00850-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/470281 (дата обращения: 19.12.2021).

11. Кувшинов, Д. Р. Основы программирования : учебное пособие для вузов / Д. Р. Кувшинов. — Москва : Издательство Юрайт, 2021. — 104 с. — (Высшее образование). — ISBN 978-5-534-07559-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/473570 (дата обращения: 19.12.2021).

12. Кузин, А. В. Программирование на языке Си : учебное пособие / А.В. Кузин, Е.В. Чумакова. — Москва : ФОРУМ : ИНФРА-М, 2020. — 143 с. — (Среднее профессиональное образование). - ISBN 978-5-00091-556-1. - Текст : электронный. - URL: https://znanium.com/catalog/product/961653 (дата обращения: 19.12.2021). – Режим доступа: по подписке.

13. Липман С. Лажойе Ж. C++ для начинающих, https://cpp.com.ru/lippman/index.html

14. Малов, А. В. Концепции современного программирования : учебное пособие для вузов / А. В. Малов, С. В. Родионов. — Москва : Издательство Юрайт, 2022. — 96 с. — (Высшее образование). — ISBN 978-5-534-14911-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/485436 (дата обращения: 19.12.2021).

15. Мезенцев А.В. Сборник задач по программированию с примерами. Иркурск: Иркутский Государственный университет, 2011. 36 с. – Режим доступа: https://newlms.magtu.ru/pluginfile.php/1575749/mod_resource/content/0/Sborn_Zadach_po_programm_s_primerami.pdf

16. Мясников Е.В., Попов А.Б. Шаблоны в C++: электронные методические указания к лабораторной работе. Самара, 2011. 24 с. Режим доступа: http://repo.ssau.ru/bitstream/Metodicheskie-ukazaniya/Shablony-v-C-Elektronnyi-resurs-elektron-metod-ukazaniya-k-lab-rabote-5-53577/1/%D0%9C%D1%8F%D1%81%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2%20%D0%95.%D0%92.%20%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B%20%D0%B2%20C%2B%2B.pdf

17. Немцова, Т. И. Программирование на языке высокого уровня. Программирование на языке C++ : учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев ; под ред. Л.Г. Гагариной. — Москва : ФОРУМ : ИНФРА-М, 2021. — 512 с. + Доп. материалы [Электронный ресурс]. — (Среднее профессиональное образование). - ISBN 978-5-8199-0699-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/1172261 (дата обращения: 19.12.2021). – Режим доступа: по подписке.

18. Огнева, М. В. Программирование на языке C++: практический курс : учебное пособие для вузов / М. В. Огнева, Е. В. Кудрина. — Москва : Издательство Юрайт, 2021. — 335 с. — (Высшее образование). — ISBN 978-5-

534-05123-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/473054 (дата обращения: 19.12.2021).

19. Сборник задач по программированию. – Одесса: ОНАС им. А.С. Попова, 2011. – 212 с.. – Режим доступа: https://vk.com/doc137181225_441506443?hash=71e6674d82b9751ea3&dl=78b2d4c67b2ce5156e

20. Торчинский, В. Е. Практикум по программированию : учебное пособие / В. Е. Торчинский, А. Н. Калитаев, В. Д. Тутарова. - Магнитогорск : МГТУ, 2013. - 1 электрон. опт. диск (CD-ROM). - Загл. с титул. экрана. - URL: https://magtu.informsystema.ru/uploader/fileUpload?name=916.pdf&show=dcatalogues/1/1118903/916.pdf&view=true (дата обращения: 08.12.2021). - Макрообъект. - Текст : электронный. - Сведения доступны также на CD-ROM.

21. Трофимов, В. В. Алгоритмизация и программирование : учебник для вузов / В. В. Трофимов, Т. А. Павловская ; под редакцией В. В. Трофимова. — Москва : Издательство Юрайт, 2021. — 137 с. — (Высшее образование). — ISBN 978-5-534-07834-3. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/471125 (дата обращения: 19.12.2021).

22. Черпаков, И. В. Основы программирования : учебник и практикум для вузов / И. В. Черпаков. — Москва : Издательство Юрайт, 2021. — 219 с. — (Высшее образование). — ISBN 978-5-9916-9983-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/469570 (дата обращения: 19.12.2021).

Учебное текстовое электронное издание

Гаврилова Ирина Викторовна

ПРОГРАММИРОВАНИЕ

Практикум

Ответственность за содержание возлагается на автора
Издается полностью в авторской редакции

1,29 Мб
1 электрон. опт. диск

г. Магнитогорск, 2022 год
ФГБОУ ВО «МГТУ им. Г.И. Носова»
Адрес: 455000, Россия, Челябинская область, г. Магнитогорск,
пр. Ленина 38

ФГБОУ ВО «Магнитогорский государственный
технический университет им. Г.И. Носова»
Кафедра бизнес-информатики и информационных технологий