

Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра вычислительных методов и программирования

В. Л. Бусько, А. А. Навроцкий

**ОСНОВЫ АЛГОРИТМИЗАЦИИ
И ПРОГРАММИРОВАНИЯ
В СРЕДЕ VISUAL C++**

Лабораторный практикум по курсу
«Основы алгоритмизации и программирования»
для студентов 1 – 2-го курсов всех специальностей БГУИР

Минск БГУИР 2009

УДК 681.3.06 (075.8)
ББК 32.973.26-018.1 я73
Б92

Бусько, В. Л.

Б92

Основы алгоритмизации и программирования в среде Visual C++ :
лаб. практикум по курсу «Основы алгоритмизации и программирования»
для студ. 1 – 2-го курсов всех спец. БГУИР / В. Л. Бусько, А. А. Навроц-
кий. – Минск : БГУИР, 2009. – 66 с. : ил.

ISBN 978-985-488-374-8

Приведено 16 лабораторных работ на языке C++ в среде Microsoft Visual Studio
с примерами выполнения; представлены индивидуальные задания; дана необходимая
справочная информация.

УДК 681.3.06 (075.8)
ББК 32.973.26-018.1 я73

ISBN 978-985-488-374-8

© Бусько В. Л., Навроцкий А. А., 2009
© УО «Белорусский государственный университет
информатики и радиоэлектроники», 2009

СОДЕРЖАНИЕ

Лабораторная работа №1. Среда программирования Visual C++.	
Программирование линейных алгоритмов	5
1.1. Консольный режим работы среды Visual C++ 6.0	5
1.2. Функции библиотеки math.lib	6
1.3. Пример выполнения работы.....	7
1.4. Индивидуальные задания	8
Лабораторная работа №2. Программирование разветвляющихся алгоритмов...	10
2.1. Логические операции и операции сравнения	10
2.2. Приоритет операций в C++	10
2.3. Оператор условной передачи управления if.....	10
2.4. Оператор множественного выбора switch	11
2.5. Пример выполнения работы.....	11
2.6. Индивидуальные задания	12
Лабораторная работа №3. Программирование циклических алгоритмов.....	14
3.1. Оператор цикла for.....	14
3.2. Оператор цикла while	14
3.3. Оператор цикла do	14
3.4. Отладка программы	14
3.5. Пример выполнения работы.....	15
3.6. Индивидуальные задания	16
Лабораторная работа №4. Программирование с использованием одномерных массивов	18
4.1. Одномерные статические массивы	18
4.2. Пример выполнения работы.....	18
4.3. Индивидуальные задания	19
Лабораторная работа №5. Указатели. Программирование с использованием динамических двумерных массивов	20
5.1. Объявление указателя.....	20
5.2. Операции над указателями	20
5.3. Создание двумерного динамического массива	21
5.4. Пример выполнения работы.....	21
5.5. Индивидуальные задания	22
Лабораторная работа №6. Программирование с использованием строк	24
6.1. Объявление строк.....	24
6.2. Функции для работы со строками.....	24
6.3. Пример выполнения работы.....	25
6.4. Индивидуальные задания	26
Лабораторная работа №7. Программирование с использованием структур	27
7.1. Объявление структур	27
7.2. Пример выполнения работы.....	27
7.3. Индивидуальные задания	29

Лабораторная работа №8. Программирование с использованием функций	31
8.1. Объявление функции	31
8.2. Передача параметров	31
8.3. Перегрузка функций и указатель на функцию	32
8.4. Пример выполнения работы	32
8.5. Индивидуальные задания	33
Лабораторная работа №9. Программирование с использованием рекурсии.....	35
9.1. Понятие рекурсии	35
9.2. Пример выполнения работы	35
9.3. Индивидуальные задания.	36
Лабораторная работа №10. Программирование с использованием файлов.....	38
10.1. Организация работы с файлами	38
10.2. Функции для работы с файлами.....	38
10.3. Пример выполнения работы	40
10.4. Индивидуальные задания	44
Лабораторная работа №11. Сортировка по ключу одномерных массивов структур	46
11.1. Сортировка массивов	46
11.2. Пример выполнения работы	47
11.3. Индивидуальные задания	48
Лабораторная работа №12. Поиск по ключу в одномерном массиве структур	49
12.1. Поиск в массиве	49
12.2. Индивидуальные задания.	50
Лабораторная работа №13. Программирование с использованием однонаправленных списков типа «стек»	51
13.1. Работа со стеками.....	51
13.2. Индивидуальные задания	52
Лабораторная работа №14. Программирование с использованием однонаправленных списков типа «очередь»	53
14.1. Работа с однонаправленными списками	53
14.2. Индивидуальные задания.	54
Лабораторная работа №15. Программирование с использованием двусвязанных списков	55
15.1. Очереди на основе двусвязанных списков.	55
15.2. Индивидуальные задания	58
Лабораторная работа №16. Программирование с использованием древовидных структур данных	60
16.1. Основные операции с бинарным деревом поиска.....	60
16.2. Индивидуальные задания	64
Литература	65

ЛАБОРАТОРНАЯ РАБОТА №1

СРЕДА ПРОГРАММИРОВАНИЯ VISUAL C++.

ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

1.1. Консольный режим работы среды Visual C++ 6.0

Программа, создаваемая в среде Visual C++, всегда оформляется в виде отдельного проекта. Проект (*project*) – это набор взаимосвязанных исходных файлов, предназначенных для решения определенной задачи, компиляция и компоновка которых позволяет получить выполняемую программу. В проект входят как файлы, непосредственно создаваемые программистом, так и файлы, которые автоматически создает и редактирует среда программирования.

Для **создания нового проекта** необходимо:

- выбрать **File – New**;
- в открывшемся окне на закладке **Projects** выбрать тип проекта **Win32**

Console Application;

- в поле **Project Name** ввести имя проекта, например *mylab1*;
- в поле **Location** ввести имя каталога, в котором будет размещен проект и полный путь к нему, например *D:\WORK\mylab1*. Каталог также можно выбрать, используя диалоговое окно **Choose Directory**, для чего надо щелкнуть мышью по кнопке ... ;

- указать тип создаваемого проекта – **Win32 Console Application**;

- щелкнуть мышью по кнопке **OK**;

• в открывшемся окне мастера приложений **Win32 Console Application – Step 1 of 1** выбрать **An empty project** (пустой проект) и щелкнуть по кнопке **Finish**;

- в открывшемся окне **New Project Information** (информация о новом проекте) щелкнуть мышью по кнопке **OK**.

Для работы с консольным приложением необходимо создать новый или добавить существующий файл с текстом программы.

Для **создания нового файла** необходимо:

- выбрать **File – New**;

- в открывшемся окне на закладке **Files** выбрать тип файла **C++ Source File**;

- в поле **File name:** ввести имя файла. Для удобства желательно ввести имя, совпадающее с именем проекта, например *mylab1*;

- щелкнуть мышью по кнопке **OK**.

Для **добавления в проект уже существующего файла** с текстом программы необходимо:

- скопировать имеющийся файл (расширение **cpp**) в рабочую папку проекта;

- в окне **Workspace**, закладка **FileView**, щелкнуть правой кнопкой мыши по папке **Source Files**;

- в открывшемся диалоговом окне **Insert Files...** выбрать добавляемый файл и щелкнуть мышью по кнопке **OK**.

В папке проекта, как правило, размещено пять файлов и одна вложенная папка. Файлы имеют следующее назначение.

Файл с расширением **dsw** (например *mylab1.dsw*) – файл проекта, который объединяет все входящие в проект файлы.

Файл с расширением **dsp** (например *mylab1.dsp*) предназначен для построения отдельного проекта или подпроекта.

Файл с расширением **opt** (например *mylab1.opt*) содержит все настройки данного проекта.

Файл с расширением **ncb** (например *mylab1.ncb*) – служебный файл.

Файл с расширением **cpp** (например *mylab1.cpp*) – файл текста программы.

1.2. Функции библиотеки *math.lib*

Функции для расчета математических выражений находятся в библиотеке *math.lib* (подключение библиотеки: **#include math.h**). Все аргументы в тригонометрических функциях задаются в радианах. Параметры и аргументы всех остальных функций имеют тип **double** (кроме *abs(x)*).

Математическая функция	Функция библиотеки <i>math.lib</i>	Описание
$ x $	abs(x)	Вычисление абсолютного значения (только для целых чисел!)
$\arccos(x)$	acos(x)	Вычисление значения арккосинуса x
$\arctg(x)$	atan(x)	Вычисление значения арктангенса x
$\arctg(x/y)$	atan2(x,y)	Вычисление значения арктангенса двух аргументов x и y
Округление к большему	ceil(x)	Функция возвращает действительное значение, соответствующее наименьшему целому числу, которое больше или равно x
$\cos(x)$	cos(x)	Вычисление косинуса x
$\operatorname{ch}(x) = (e^x + e^{-x})/2$	cosh(x)	Вычисление косинуса гиперболического x
e^x	exp(x)	Вычисление экспоненты числа x
$ x $	fabs(x)	Вычисление абсолютного значения x
Округление к меньшему	floor(x)	Функция возвращает действительное значение, соответствующее наибольшему целому числу, которое меньше или равно x
Остаток от деления x на y	fmod(x,y)	Функция возвращает действительное значение, соответствующее остатку от целочисленного деления x на y
$\ln(x)$	log(x)	Вычисление натурального логарифма x
$\lg_{10}(x)$	log10(x)	Вычисление десятичного логарифма x
x^y	pow(x, y)	Возведение x в степень y
$\sin(x)$	sin(x)	Вычисление синуса x

$\text{sh}(x) = (e^x - e^{-x})/2$	sinh(x)	Вычисление синуса гиперболического x
\sqrt{x}	sqrt(x)	Вычисление квадратного корня x
$\text{tg}(x)$	tan(x)	Вычисление тангенса x
$\text{tgh}(x)$	tanh(x)	Вычисление тангенса гиперболического x

1.3. Пример выполнения работы

Условие: написать программу для вычисления линейного арифметического выражения

$$h = \frac{x^{2y} + e^{y-1}}{1 + x|y - \text{tg}z|} + 10 \cdot \sqrt[3]{x} - \ln(z).$$

При $x = 2.45$, $y = -0.423 \times 10^{-2}$, $z = 1.232 \times 10^3$ ответ $h = 6.9465$.

Текст программы:

```
#include <iostream.h>
#include <math.h>
int main ()
{
    double x,y,z,a,b,c,h;
    cout << "Vvedite x: ";
    cin >> x;
    cout << "Vvedite y: ";
    cin >> y;
    cout << "Vvedite z: ";
    cin >> z;
    a = pow(x,2*y)+exp(y-1);
    b = 1+x*fabs(y-tan(z));
    c = 10*pow(x,1/3.)-log(z);
    h = a/b+c;
    cout << "Result h= " << h << endl;
    return 0;
}
```

Для компиляции, компоновки и запуска программы на выполнение используются следующие пункты подменю Build:

Compile (Ctrl+F7) – компиляция выбранного файла. Результаты компиляции выводятся в окно Output.

Build (F7) – компоновка проекта. Компилируются все файлы, в которых произошли изменения с момента последней компоновки. Если компоновка прошла без ошибок, то среда программирования создаст исполняемый файл с расширением **exe**, который можно будет запустить на выполнение.

Rebuild All – перекомпоновка проекта. Компилируются все файлы проекта независимо от того, были ли в них произведены изменения или нет.

Execute (Ctrl+F5) – выполнение исполняемого файла, созданного в результате компоновки проекта. Для файлов, в которые были внесены изменения, выполняется перекомпилирование и перекомпоновка.

Если в процессе компиляции были обнаружены синтаксические ошибки, то выводится соответствующее сообщение. В этом случае необходимо последовательно исправлять ошибки и компилировать проект снова. Если синтаксических ошибок нет, но результат выполнения программы неверный, необходимо искать логические ошибки. Для этого следует использовать встроенный в систему отладчик (см. лаб. работу №2).

После окончания работы проект можно закрыть, выбрав **File – Close Workspace**, или закрыть приложение MVC++.

Для открытия сохраненного ранее проекта необходимо выбрать **File – Open Workspace...** В открывшемся диалоговом окне выбрать папку проекта и открыть в ней файл с расширением **dsw**.

1.4. Индивидуальные задания

Вычислить значение выражения при заданных исходных данных. Сравнить полученное значение с указанным правильным результатом.

$$1. s = \frac{2 \cos\left(x - \frac{2}{3}\right)}{\frac{1}{2} + \sin^2 y} \left(1 + \frac{z^2}{3 - z^2/5}\right).$$

При $x = 14.26$; $y = -1.22$; $z = 3.5 \times 10^{-2}$. Ответ: $s = 0.749155$.

$$2. s = \frac{\sqrt[3]{9 + (x - y)^2}}{x^2 + y^2 + 2} - e^{|x-y|} \operatorname{tg}^3 z.$$

При $x = -4.5$; $y = 0.75 \times 10^{-4}$; $z = -0.845 \times 10^2$. Ответ: $s = -3.23765$.

$$3. s = \frac{1 + \sin^2(x + y)}{\left|x - \frac{2y}{1 + x^2 y^2}\right|} x^{|y|} + \cos^2\left(\operatorname{arctg} \frac{1}{z}\right).$$

При $x = 3.74 \times 10^{-2}$; $y = -0.825$; $z = 0.16 \times 10^2$. Ответ: $s = 1.05534$.

$$4. s = |\cos x - \cos y|^{(1+2\sin^2 y)} \left(1 + z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4}\right).$$

При $x = 0.4 \times 10^4$; $y = -0.875$; $z = -0.475 \times 10^{-3}$. Ответ: $s = 1.98727$.

$$5. s = \ln\left(y^{-\sqrt{|x|}}\right) \left(x - \frac{y}{2}\right) + \sin^2(\operatorname{arctg}(z)).$$

При $x = -15.246$; $y = 4.642 \times 10^{-2}$; $z = 21$. Ответ: $s = -182.038$.

$$6. s = \sqrt{10\left(\sqrt[3]{x} + x^{y+2}\right)} \left(\arcsin^2 z - |x - y|\right).$$

При $x = 16.55 \times 10^{-3}$; $y = -2.75$; $z = 0.15$. Ответ: $s = -40.6307$.

$$7. s = 5 \operatorname{arctg}(x) - \frac{1}{4} \arccos(x) \frac{x + 3|x - y| + x^2}{|x - y|z + x^2}.$$

При $x = 0.1722$; $y = 6.33$; $z = 3.25 \times 10^{-4}$. Ответ: $s = -205.306$.

$$8. s = \frac{e^{|x-y|} |x-y|^{x+y}}{\operatorname{arctg}(x) + \operatorname{arctg}(z)} + \sqrt[3]{x^6 + \ln^2 y}.$$

При $x = -2.235 \times 10^{-2}$; $y = 2.23$; $z = 15.221$. Ответ: $s = 39.3741$.

$$9. s = \left| x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}} \right| + (y-x) \frac{\cos y - \frac{z}{(y-x)}}{1 + (y-x)^2}.$$

При $x = 1.825 \times 10^2$; $y = 18.225$; $z = -3.298 \times 10^{-2}$. Ответ: $s = 1.21308$.

$$10. s = 2^{-x} \sqrt{x + 4\sqrt{|y|}} \sqrt[3]{e^{x-1/\sin z}}.$$

При $x = 3.981 \times 10^{-2}$; $y = -1.625 \times 10^3$; $z = 0.512$. Ответ: $s = 1.26185$.

$$11. s = y^{\sqrt[3]{|x|}} + \cos^3(y) \frac{|x-y| \left(1 + \frac{\sin^2 z}{\sqrt{x+y}} \right)}{e^{|x-y|} + \frac{x}{2}}.$$

При $x = 6.251$; $y = 0.827$; $z = 25.001$. Ответ: $s = 0.712122$.

$$12. s = 2^{(y^x)} + (3^x)^y - \frac{y \left(\operatorname{arctg} z - \frac{1}{3} \right)}{|x| + \frac{1}{y^2 + 1}}.$$

При $x = 3.251$; $y = 0.325$; $z = 0.466 \times 10^{-4}$. Ответ: $s = 4.23655$.

$$13. s = \frac{\sqrt[4]{y + \sqrt[3]{x-1}}}{|x-y| (\sin^2 z + \operatorname{tg} z)}.$$

При $x = 17.421$; $y = 10.365 \times 10^{-3}$; $z = 0.828 \times 10^5$. Ответ: $s = 0.330564$.

$$14. s = \frac{y^{x+1}}{\sqrt[3]{|y-2|} + 3} + \frac{x + \frac{y}{2}}{2|x+y|} (x+1)^{-1/\sin z}.$$

При $x = 12.3 \times 10^{-1}$; $y = 15.4$; $z = 0.252 \times 10^3$. Ответ: $s = 82.8256$.

$$15. s = \frac{x^{y+1} + e^{y-1}}{1 + x|y - \operatorname{tg} z|} (1 + |y-x|) + \frac{|y-x|^2}{2} - \frac{|y-x|^3}{3}.$$

При $x = 2.444$; $y = 0.869 \times 10^{-2}$; $z = -0.13 \times 10^3$. Ответ: $s = -0.498707$.

ЛАБОРАТОРНАЯ РАБОТА №2 ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

2.1. Логические операции и операции сравнения

Операции сравнения применяются при работе с двумя операндами и возвращают true (1), если результат сравнения – истина, и false (0), если результат сравнения – ложь. В языке Си определены следующие операции сравнения: < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), != (не равно), = (равно).

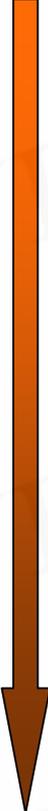
Логические операции работают с операндами скалярных типов и возвращают результат булева типа. Существует три логические операции:

- ! – отрицание, или логическое **НЕ**;
- && – логическое **И**;
- || – логическое **ИЛИ**.

2.2. Приоритет операций в C++

Тип операции	Операторы
Разрешения области действия	::
Другие	[], (), «.» (точка),
Унитарные	&, +, -, !, префикс ++ и --
Унитарные	*, ->
Арифметические	*, /, %
Арифметические	+, -
Унитарные	постфикс ++ и --
Поразрядный сдвиг	<<, >>
Сравнение	>, <, >=, <=
Сравнение	==, !=
Поразрядные логические	&
Поразрядные логические	^
Поразрядные логические	
Логические	&&
Логические	
Условная	?:
Присваивания	=, *=, /=, %=, +=, -=, <<=, >>=, &=, ^=, =
Последовательность	<<,» (запятая)

Уменьшение приоритета



2.3. Оператор условной передачи управления if

Формат оператора выбора:

if (логическое выражение) оператор 1;
else оператор 2;

Если логическое выражение истинно, то выполняется оператор 1, иначе – оператор 2.

2.4. Оператор множественного выбора switch

Общая форма оператора следующая:

```
switch(переменная выбора) {  
    case const 1: операторы 1; break;  
    ...  
    case const N: операторы N; break;  
    default: операторы N+1;  
}
```

При использовании оператора **switch** сначала анализируется *переменная выбора* и проверяется, совпадает ли ее значение со значением одной из констант. При совпадении выполняются операторы этого **case**. Конструкция **default** (может отсутствовать) выполняется, если результат выражения не совпал ни с одной из констант.

2.5. Пример выполнения работы

Условие. Вычислить значение выражения $s = \begin{cases} |f(x)| + \ln(y), & |xy| > 10, \\ e^{f(x)+y}, & 5 < |xy| \leq 10, \\ \sin(x) + \operatorname{tg}(y), & |xy| = 5. \end{cases}$

При выполнении задания предусмотреть выбор вида функции $f(x)$: $\operatorname{sh}(x)$, x^2 или e^x .
Текст программы:

```
#include <iostream.h>  
#include <math.h>  
int main()  
{  
    double x,y,f,a,s;  
    int k;  
    cout << "Vvedite x "; cin >> x;  
    cout << "Vvedite y "; cin >> y;  
    cout << "Viberite f: 1 - sh(x), 2 - x^2, 3 - exp(x) "; cin >> k;  
    switch(k)  
    {  
        case 1: f=sinh(x); break;  
        case 2: f=pow(x,2); break;  
        case 3: f=exp(x); break;  
        default: cout << "Ne vubrana funkciya "; return 1;  
    }  
    a=fabs(x*y);  
    if (a<5) {
```

```

cout << "Net rezultata" << endl;
return 1;
} else
if (a>10) s=fabs(f)+log(y);
else
if (a<=10 && a>5) s=exp(f+y);
else s=sin(x)+tan(y);
cout << "RESULT = " << s << endl;
return 0;
}

```

2.6. Индивидуальные задания

При выполнении задания предусмотреть выбор вида функции $f(x)$: $\text{sh}(x)$, x^2 или e^x . Предусмотреть вывод информации о выбранной ветви вычислений.

$$\begin{array}{ll}
1. \quad a = \begin{cases} (f(x) + y)^2 - \sqrt[3]{|f(x)|}, & xy > 0 \\ (f(x) + y)^2 + \sin(x), & xy < 0 \\ (f(x) + y)^2 + y^3, & xy = 0. \end{cases} & 2. \quad b = \begin{cases} \ln(f(x)) + \sqrt[3]{|f(x)|}, & x/y > 0 \\ \ln|f(x)/y| \cdot (x+y)^3, & x/y < 0 \\ (f(x)^2 + y)^3, & \text{иначе.} \end{cases} \\
3. \quad c = \begin{cases} f(x)^2 + \sqrt[3]{y} + \sin(y), & x - y = 0 \\ (f(x) - y)^2 + \ln(x), & x - y > 0 \\ (y - f(x))^2 + \text{tg}(y), & x - y < 0. \end{cases} & 4. \quad d = \begin{cases} \sqrt[3]{|f(x) - y|} + \text{tg}(f(x)), & x > y \\ (y - f(x))^3 + \cos(f(x)), & y < x \\ (y + f(x))^2 + x^3, & y = x. \end{cases} \\
5. \quad e = \begin{cases} y\sqrt{f(x)} + 3\sin(x), & x > y \\ x\sqrt{|f(x)|}, & x < y \\ \sqrt[3]{|f(x)|} + x^3/y, & \text{иначе.} \end{cases} & 6. \quad g = \begin{cases} e^{f(x)-|y|}, & 0.5 < xy < 10 \\ \sqrt[3]{|f(x) + y|}, & 0.1 < xy < 0.5 \\ 2f(x)^2, & \text{иначе.} \end{cases} \\
7. \quad s = \begin{cases} e^{f(x)}, & 1 < xb < 10 \\ \sqrt[3]{|f(x) + 4y|}, & 12 < xb < 40 \\ y \cdot f(x)^2, & \text{иначе.} \end{cases} & 8. \quad b = \begin{cases} (f(x)^2 + y)^3, & x/y < 0 \\ \ln|f(x)/y| + x/y, & x/y > 0 \\ \sqrt[3]{|\sin(y)|}, & \text{иначе.} \end{cases} \\
9. \quad l = \begin{cases} 2f(x)^3 + 3y^2, & x > |y| \\ |f(x) - y|, & 3 < x < |y| \\ \sqrt[3]{|f(x) - y|}, & \text{иначе.} \end{cases} & 10. \quad k = \begin{cases} \ln(|f(x)| + |y|), & |xy| > 10 \\ e^{f(x)+y}, & |xy| < 10 \\ \sqrt[3]{|f(x)|} + y, & |xy| = 10. \end{cases}
\end{array}$$

$$11. \quad b = \begin{cases} \operatorname{tg}(f(x)) + x/\sqrt[3]{y}, & xy > 0 \\ \ln|f(x)^2 \cdot y|, & xy < 0 \\ f(x)^2 + \sin^2(y), & \text{иначе.} \end{cases}$$

$$12. \quad b = \begin{cases} \operatorname{tg}(x) + f(x)^2, & y > 2x \\ |f(x) + y|^3, & y < 2x \\ \sqrt[3]{x} \cdot \sin(x), & \text{иначе.} \end{cases}$$

$$13. \quad b = \begin{cases} (f(x) + \ln(y))^3, & x/y > 0 \\ 2/3 + \ln(|\sin(y)|), & x/y < 0 \\ \sqrt[3]{f(x)^2 + y}, & \text{иначе.} \end{cases}$$

$$14. \quad b = \begin{cases} \ln(f(x))^3, & x^3 > 0 \\ \operatorname{tg}(x^3) + f(x), & x^3 < 0 \\ \sqrt[3]{|y^3 - x^2|}, & \text{иначе.} \end{cases}$$

$$15. \quad b = \begin{cases} (x^2 + f(x)^2)/y, & f(x) > 0 \\ \ln|f(x)^3| + \cos(f(x)), & f(x) < 0 \\ \sqrt[3]{\sin^2(y)}, & \text{иначе.} \end{cases}$$

ЛАБОРАТОРНАЯ РАБОТА №3

ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

3.1. Оператор цикла *for*

Общий вид оператора:

```
for (инициализирующее выражение; условие;  
      инкрементирующее выражение)  
{  
  тело цикла  
}
```

Инициализирующее выражение выполняется только один раз в начале выполнения цикла и, как правило, инициализирует счетчик цикла.

Условие содержит операцию отношения, которая выполняется в начале каждого цикла. Если условие равно 1 (true), то цикл повторяется, иначе выполняется следующий за телом цикла оператор.

Инкрементирующее выражение, как правило, предназначено для изменения значения счетчика цикла. Модификация счетчика происходит после каждого выполнения тела цикла.

3.2. Оператор цикла *while*

Оператор цикла с предусловием. Общий вид оператора:

```
while (условие)  
{  
  тело цикла  
}
```

Организует повторение операторов *тела цикла* до тех пор, пока *условие* истинно.

3.3. Оператор цикла *do*

Оператор цикла с постусловием. Общий вид оператора:

```
do {  
  тело цикла  
}  
while (условие);
```

Организует повторение операторов *тела цикла* до тех пор, пока *условие* истинно.

3.4. Отладка программы

Для поиска логических ошибок используется встроенный отладчик.

Для пошагового выполнения программы необходимо нажимать клавишу **F10**. При каждом нажатии выполняется текущая строка. Если необходимо пошагово проверить текст вызываемой функции, то следует нажать **F11**. Для досрочного выхода из функции нажать **Shift+F11**. Если необходимо начать отладку с определенного места программы, то надо установить курсор в соответствующую строку программы и нажать **Ctrl+F10**.

Другим способом отладки является установка *точек прерывания* программы. Для этого надо поместить курсор в нужную строку и нажать **F9**. Точка прерывания обозначается красным кружком на специальном поле, расположенном слева от окна текста программы. Для удаления точки прерывания следует в необходимой строке повторно нажать **F9**. Количество точек прерывания в программе может быть любым.

Для выполнения программы до точки прерывания необходимо нажать **F5**. Для продолжения отладки нажимается клавиша **F5** (для выполнения программы до следующей точки прерывания) или используются клавиши для пошаговой отладки.

Желтая стрелка на поле слева от окна текста программы указывает на строку, которая будет выполнена на следующем шаге отладки.

Для контроля за значениями переменных удобно использовать следующий способ: подвести указатель мыши к интересующей переменной и задержать его на несколько секунд. На экране рядом с именем переменной появится окно, содержащее текущее значение этой переменной. Кроме этого, значения переменных будут отображаться в окнах, расположенных снизу. В левом нижнем окне отображаются значения последних использованных программой переменных. В правом нижнем окне (Watch) можно задать имена переменных, значения которых необходимо контролировать.

3.5. Пример выполнения работы

Условие 1. Вычислить простое рекуррентное выражение $\sum_{k=0}^{100} -1^k \frac{x^k}{k!}$.

Перед написанием алгоритма следует получить рекуррентную формулу. Для этого рассматриваются значения слагаемых при различных k : при $k=1$ $a_1 = -1 \frac{x}{1}$; при $k=2$ $a_2 = 1 \frac{x \cdot x}{1 \cdot 2}$; при $k=3$ $a_3 = -1 \frac{x \cdot x \cdot x}{1 \cdot 2 \cdot 3}$ и т. д. Видно, что

очередное слагаемое отличается от предыдущего на множитель $-\frac{x}{k}$. Исходя из

этого формула рекуррентной последовательности имеет вид $a_k = -a_{k-1} \frac{x}{k}$. Полученная формула позволяет избавиться от многократного вычисления факториала и возведения в степень.

Текст программы:

```
...
s=a=1;
for(int i=1; i<101; i++)
{
    a *= -x/i;
    s += a;
}
```

Условие 2. Вывести на экран таблицу значений функции $Y(x) = 9^x$ и ее разложения в ряд $S(x) = 1 + \frac{\ln 9}{1}x + \dots + \frac{(\ln 9)^n}{k!}x^n$, $n = 50$, на интервале $[-3, 3]$, $h = 0.1$.

```

#include <iostream.h>
#include <iomanip.h>
#include <math.h>
int main()
{
double a,b,h,x,y,s,p;
int n,i;
cout << "Vvedite a,b,h,n" << endl;
cin >> a >> b >> h >> n;
x=a;
do
{
p=s=1;
for (i=1; i<=n; i++)
{
p *= log(9)*x/i;
s += p;
}
y=pow(9,x);
cout << setw(15) << x << setw(15) << y << setw(15) << s << endl;
x += h;
}
while (x <= b+h/2);
cout << endl;
return 0;
}

```

3.6. Индивидуальные задания

Вывести на экран таблицу значений функции $Y(x)$ и ее разложения в ряд $S(x)$ для x , изменяющегося от a до b с шагом $h = (b - a)/10$, табл. 3.1.

Таблица 3.1

№	a	b	$S(x)$	n	$Y(x)$
1	2	3	4	5	6
1	0.1	1	$x - \frac{x^3}{3!} + \dots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$	160	$\sin x$
2	0.1	1	$1 + \frac{x^2}{2!} + \dots + \frac{x^{2n}}{(2n)!}$	100	$\frac{e^x + e^{-x}}{2}$

Окончание табл. 3.1

1	2	3	4	5	6
3	0.1	1	$1 + \frac{\cos \frac{\pi}{4}}{1!} x + \dots + \frac{\cos n \frac{\pi}{4}}{n!} x^n$	120	$e^{x \cos \frac{\pi}{4}} \cos(x \sin \frac{\pi}{4})$
4	0.1	1	$1 - \frac{x^2}{2!} + \dots + (-1)^n \frac{x^{2n}}{(2n)!}$	80	$\cos x$
5	0.1	1	$1 + 3x^2 + \dots + \frac{2n+1}{n!} x^{2n}$	140	$(1 + 2x^2)e^{x^2}$
6	0.1	1	$x + \frac{x^3}{3!} + \dots + \frac{x^{2n+1}}{(2n+1)!}$	80	$\frac{e^x - e^{-x}}{2}$
7	0.1	1	$\frac{x^3}{3} - \frac{x^5}{15} + \dots + (-1)^{n+1} \frac{x^{2n+1}}{4n^2 - 1}$	120	$\frac{1+x^2}{2} \operatorname{arctg} x - \frac{x}{2}$
8	0.1	1	$1 + \frac{2x}{1!} + \dots + \frac{(2x)^n}{n!}$	100	e^{2x}
9	0.1	1	$1 + 2 \frac{x}{2} + \dots + \frac{n^2+1}{n!} \left(\frac{x}{2}\right)^n$	140	$\left(\frac{x^2}{4} + \frac{x}{2} + 1\right) e^{\frac{x}{2}}$
10	0.1	0.5	$x - \frac{x^3}{3} + \dots + (-1)^n \frac{x^{2n+1}}{2n+1}$	150	$\operatorname{arctg} x$
11	0.1	1	$1 - \frac{3}{2} x^2 + \dots + (-1)^n \frac{2n^2+1}{(2n)!} x^{2n}$	100	$\left(1 - \frac{x^2}{2}\right) \cos x - \frac{x}{2} \sin x$
12	0.1	1	$-\frac{(2x)^2}{2} + \frac{(2x)^4}{24} - \dots + (-1)^n \frac{(2x)^{2n}}{(2n)!}$	80	$2(\cos^2 x - 1)$
13	-2	-0.1	$-(1+x)^2 + \frac{(1+x)^4}{2} + \dots + (-1)^n \frac{(1+x)^{2n}}{n}$	160	$\ln \frac{1}{2+2x+x^2}$
14	0.2	0.8	$\frac{x}{3!} + \frac{4x^2}{5!} + \dots + \frac{n^2}{(2n+1)!} x^n$	120	$\frac{1}{4} \left(\frac{x+1}{\sqrt{x}} \operatorname{sh} \sqrt{x} - \operatorname{ch} \sqrt{x} \right)$
15	0.1	0.8	$\frac{x^2}{2} - \frac{x^4}{12} + \dots + (-1)^{n+1} \frac{x^{2n}}{2n(2n-1)}$	180	$x \cdot \operatorname{arctg} x - \ln \sqrt{1+x^2}$

ЛАБОРАТОРНАЯ РАБОТА №4

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ОДНОМЕРНЫХ МАССИВОВ

4.1. Одномерные статические массивы

В программе одномерный массив объявляется следующим образом:

```
тип имя массива [размер];
```

Пример объявления массива:

```
int mas[4];
```

Индексы в массиве начинаются с 0 (т. е. массив, приведенный в примере, будет содержать следующие элементы: mas[0], mas[1], mas[2] и mas[3]). Выход индекса за пределы массива не проверяется.

4.2. Пример выполнения работы

Условие 1. Удалить из одномерного массива все отрицательные элементы

```
...
for (i=0; i<n; i++)
    if (a[i]<0)
    {
        for (j=i+1; j<n; j++) a[j-1]=a[j];
        n--; i--;
    }
...

```

Условие 2. Элементы одномерных массивов X и Y упорядочены по возрастанию. Объединить элементы этих двух массивов в один массив Z так, чтобы он оказался упорядоченным по возрастанию.

```
...
k=i=j=0;
while(i<n && j<n)
{
    if (a[i]<b[j]) { c[k]=a[i]; i++; }
    else { c[k]=b[j]; j++; }
    k++;
}
while(i<n)
{
    c[k]=a[i]; i++; k++;
}
while(j<n)
{
    c[k]=b[j]; j++; k++;
}

```

4.3. Индивидуальные задания

Выполнить задание в соответствии с выданным вариантом.

1. Задан массив из k символов. Преобразовать массив следующим образом: сначала должны стоять цифры, входящие в массив, а затем все остальные символы. Взаимное расположение символов в каждой группе не должно изменяться.
2. Задан массив из k символов. Преобразовать массив следующим образом: расположить символы в обратном порядке.
3. Задан массив из k чисел. Найти число, наиболее часто встречающееся в этом массиве.
4. Задан массив из k чисел. Отсортировать элементы массива по возрастанию.
5. Задан массив из k чисел. Найти числа, входящие в массив только один раз.
6. Задан массив из k чисел. Сдвинуть элементы массива циклически на n позиций влево.
7. Задан массив из k чисел. Сдвинуть элементы массива циклически на n позиций вправо.
8. Задан массив из k чисел. Преобразовать массив следующим образом: все отрицательные элементы массива перенести в начало, а все остальные – в конец, сохранив исходное взаимное расположение как среди отрицательных, так и среди положительных элементов.
9. Задан массив из k символов. Создать два новых массива: в первый перенести все цифры из исходного массива, во второй – все остальные символы.
10. Задан массив из k символов. Определить, симметричен ли он, т. е. читается ли он одинаково слева направо и справа налево.
11. Задано два массива. Найти наименьшие среди элементов первого массива, которые не входят во второй массив.
12. Задан массив из k чисел. Определить количество инверсий в массиве (т. е. таких пар элементов, в которых большее число находится слева от меньшего).
13. Задан массив из k символов. Удалить из него повторные вхождения каждого символа.
14. Задан массив из k символов. Определить количество различных элементов в массиве.
15. Задан массив из k символов латинского алфавита. Вывести на экран в алфавитном порядке все символы, которые входят в этот массив по одному разу.

ЛАБОРАТОРНАЯ РАБОТА №5 УКАЗАТЕЛИ. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ДИНАМИЧЕСКИХ ДВУМЕРНЫХ МАССИВОВ

5.1. Объявление указателя

Для всех переменных выделяются участки памяти размером, соответствующим типу переменной. Программист имеет возможность работать непосредственно с адресами, для чего определен соответствующий тип данных – *указатель*. Указатель имеет следующий формат:

*тип *имя указателя*

Например:

int *a; double *b, *d; char *c;

Знак «звездочка» относится к имени указателя. Значение указателя соответствует первому байту участка памяти, на который он ссылается. На один и тот же участок памяти может ссылаться любое число указателей.

В языке Си существует три вида указателей:

1. Указатель на объект известного типа. Содержит адрес объекта определенного типа.
2. Указатель типа **void**. Применяется, если тип объекта заранее не определен.
3. Указатель на функцию.

5.2. Операции над указателями

Над указателями можно провести две унитарные операции:

1. **&** (**взять адрес**). Указатель получает адрес переменной. Данная операция применима к переменным, под которые выделен соответствующий участок памяти.
2. ***** (**операция разадресации**). Предназначена для доступа к величине, расположенной по данному адресу.

Над указателями можно выполнять арифметические операции сложения, инкремента, вычитания, декремента и операции сравнения. При выполнении арифметических операций с указателями автоматически учитывается размер данных, на которые он указывает.

Указатели, как правило, используются при работе с динамической памятью (*heap*, или «куча»). Для работы с динамической памятью в языке Си определены следующие функции: **malloc**, **calloc**, **realloc** и **free**.

В языке C++ для выделения и освобождения памяти определены операции **new** и **delete** соответственно. Используют две формы операций:

1. **Тип *указатель = new тип (значение)** – выделение участка памяти в соответствии с указанным типом и занесение туда указанного значения.

delete указатель – освобождение выделенной памяти.

2. **Тип *указатель = new тип[n]** – выделение участка памяти размером *n* блоков указанного типа.

delete []указатель – освобождение выделенной памяти.

5.3. Создание двумерного динамического массива

Имя любого массива рассматривается компилятором как указатель на нулевой элемент массива. Так как имя двумерного динамического массива является указателем на указатель, то сначала выделяется память под указатели, а затем под соответствующие этим указателям строки. Освобождение выделенной памяти происходит в обратном порядке:

```
double **umas2;  
umas2 = new double*[n];  
for(i=0; i<n; i++)  
    umas2[i] = new double[m];
```

...

```
for(i=0; i<n; i++)  
    delete []umas2[i];  
delete []umas2;  
umas2=NULL;
```

5.4. Пример выполнения работы

Условие 1. Найти минимальный и максимальный элементы матрицы и их координаты.

```
min=max=a[0][0];  
imin=jmin=imax=jmax=0;  
for (i=0; i<n; i++)  
    for (j=0; j<m; j++)  
    {  
        if (a[i][j]<min) { min=a[i][j]; imin=i; jmin=j; }  
        else  
        if (a[i][j]>max) { max=a[i][j]; imax=i; jmax=j; }  
    }
```

Условие 2. Упорядочить строки матрицы по неубыванию их максимальных элементов.

```
for (i=0; i<n; i++)  
    {  
        b[i]=a[i][0];  
        for (j=1; j<m; j++)  
            if (a[i][j]>b[i]) b[i]=a[i][j];  
    }  
  
for (i=0; i<n-1; i++)  
    for (j=i+1; j<n; j++)  
        if (b[i]>b[j])
```

```

{
  t=b[i];
  b[i]=b[j];
  b[j]=t;
  for (k=0; k<m; k++)
  {
    t=a[i][k];
    a[i][k]=a[j][k];
    a[j][k]=t;
  }
}

```

5.5. Индивидуальные задания

В работе память для массива должна выделяться динамически. На экран выводить исходные данные и результат.

1. Задана матрица размером $N \times M$. Получить массив B , присвоив его k -му элементу значение 0, если все элементы k -го столбца матрицы нулевые, и значение 1 – в противном случае.

2. Задана матрица размером $N \times M$. Получить массив B , присвоив его k -му элементу значение 1, если элементы k -й строки матрицы упорядочены по убыванию, и значение 0 – в противном случае.

3. Задана матрица размером $N \times M$. Получить массив B , присвоив его k -му элементу значение 1, если k -я строка матрицы симметрична, и значение 0 – в противном случае.

4. Задана матрица размером $N \times M$. Определить количество «особых» элементов матрицы, считая элемент «особым», если он больше суммы остальных элементов своего столбца.

5. Задана матрица размером $N \times M$. Определить количество «особых» элементов матрицы, считая элемент «особым», если в строке слева от него находятся элементы, меньшие его, а справа – большие.

6. Задана символьная матрица размером $N \times M$. Определить количество различных элементов матрицы (т.е. повторяющиеся элементы считать один раз).

7. Дана матрица размером $N \times M$. Упорядочить ее строки по возрастанию их первых элементов.

8. Дана матрица размером $N \times M$. Упорядочить ее строки по возрастанию суммы их элементов.

9. Дана матрица размером $N \times M$. Упорядочить ее столбцы по возрастанию их наименьших элементов.

10. Определить, является ли заданная квадратная матрица n -го порядка симметричной относительно побочной диагонали.

11. Для матрицы размером $N \times M$ вывести на экран все седловые точки. Элемент матрицы называется седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце или наоборот.

12. В матрице размером $N \times M$ переставить строки так, чтобы на главной диагонали матрицы были расположены элементы, наибольшие по абсолютной величине.

13. В матрице размером $N \times M$ найти максимальный среди элементов, лежащих ниже побочной диагонали, и минимальный среди элементов, лежащих выше главной диагонали.

14. В матрице размером $N \times M$ поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением.

15. Из матрицы размером $N \times M$ получить матрицу размером $(N - 1) \times (M - 1)$ путем удаления из исходной матрицы строки и столбца, на пересечении которых расположен элемент с наибольшим по модулю значением.

ЛАБОРАТОРНАЯ РАБОТА №6 ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СТРОК

6.1. Объявление строк

Объявление строки аналогично объявлению массива:

`char имя строки [размер]`

Важной особенностью является то, что строка должна обязательно заканчиваться нулевым символом `'\0'` – (нуль-терминатор). Длина строки равна количеству символов плюс нулевой символ.

6.2. Функции для работы со строками

Функции для работы со строками содержатся в библиотеке `string.lib` (подключение: `#include string.h`).

Наиболее часто применяются следующие функции:

char *strcpy(st1, st2) – копирует содержимое строки `st2`, включая нулевой символ, в строку `st1`.

char *strcat(st1, st2) – добавляет справа к строке `st1` содержимое строки `st2`.

int strcmp(st1, st2) – сравнивает содержимое строк `st2` и `st1`. Если `st1 < st2`, то результат равен `-1`, если `st1 = st2` – результат равен нулю, если `st1 > st2` – результат равен `1`.

char *strstr(st1, st2) – возвращает указатель на первое появление подстроки `st2` в строке `st1`.

char *strchr(st, sh) – возвращает указатель на первое появление символа `ch` в строке `st`.

char *strtok(st1, st2) – возвращает указатель на лексему, находящуюся в строке `st1`. При первом вызове функция возвращает указатель на первый символ в `st1`, а после первой лексемы устанавливает нулевой символ. При последующих вызовах функции со значением `NULL` в качестве первого аргумента указатель аналогичным образом переходит к следующим лексемам. После того как закончились все лексемы, указатель устанавливается в `NULL`.

int strlen(st) – возвращает длину строки `st`.

char *strrev(st) – изменяет порядок следования символов в строке на противоположный.

char *strdup(st); – дублирует строку `st`.

char *strlwr(st) – конвертирует символы строки `st` к нижнему регистру.

char *strupr(st) – конвертирует символы строки `st` к верхнему регистру.

int atoi(st) – преобразует строку `st` в число целого типа (`int`).

double atof(st) – преобразует строку `st` в число действительного типа.

char *itoa(a, st, base) – преобразует число целого типа `a` в строку `st` (`base` – основание системы счисления).

char *gcvt(a, dec, st); – преобразует число действительного типа `a` в строку `st`. Значение `dec` указывает на число десятичных разрядов (не более 18).

6.3. Пример выполнения работы

Условие 1. Выделить и вывести на экран все слова произвольной строки. Слова отделяются друг от друга одним или несколькими пробелами.

```
char st[100], sl[100];
int k=0, i;
gets(st);
strcat(st, " ");
int n=strlen(st);
    if (n<2) return 1;
sl[0]='\0';
for (i=0; i<n; i++)
    if (st[i] != ' ')
    {
        sl[k]=st[i];
        sl[k+1]='\0';
        k++;
    }
    else
    {
        if (strlen(sl)>0) puts(sl);
        sl[0]='\0';
        k=0;
    }
}
```

Условие 2. Определить, является ли строка палиндромом, т.е. читается ли она слева направо так же, как и справа налево.

```
char st[80]="A roza upala na lapu Azora";
int i,j;
bool bl=true;
strlwr(st);
i=0; j=strlen(st)-1;

while (i<=j) {
    while (st[i]==' ') i++;
    while (st[j]==' ') j--;
    if (st[i] !=st[j])
    {
        bl=false;
        break;
    }
    i++; j--;
}
if (bl) cout << "Palindrom" << endl;
else cout << "Ne palindrom" << endl;
```

6.4. Индивидуальные задания

1. Дана строка, состоящая из групп нулей и единиц. Каждая группа отделяется от другой одним или несколькими пробелами. Найти количество групп с пятью символами.
2. Дана строка, состоящая из групп нулей и единиц. Найти и вывести на экран самую короткую группу.
3. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество символов в самой длинной группе.
4. Дана строка, состоящая из групп нулей и единиц. Найти и вывести на экран группы с четным количеством символов.
5. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество единиц в группах с нечетным количеством символов.
6. Дана строка, состоящая из букв, цифр, запятых, точек, знаков «+» и «-». Выделить подстроку, которая соответствует записи целого числа.
7. Дана строка символов, состоящая из букв, цифр, запятых, точек, знаков «+» и «-». Выделить подстроку, которая соответствует записи вещественного числа с фиксированной точкой.
8. Дана строка символов, состоящая из букв, цифр, запятых, точек, знаков «+» и «-». Выделить подстроку, которая соответствует записи вещественного числа с плавающей точкой.
9. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести на экран числа этой строки в порядке возрастания их значений.
10. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести четные числа этой строки.
11. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Вывести на экран слова этого текста в порядке, соответствующем латинскому алфавиту.
12. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Вывести на экран порядковый номер слова, накрывающего k -ю позицию (если на k -ю позицию попадает пробел, то номер предыдущего слова).
13. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Разбить исходную строку на две подстроки, причем первая длиной k символов (если на k -ю позицию попадает слово, то его следует отнести ко второй строке).
14. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Вывести на экран порядковый номер слова максимальной длины и номер позиции в строке, с которой оно начинается.
15. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Вывести на экран порядковый номер слова минимальной длины и количество символов в этом слове.

ЛАБОРАТОРНАЯ РАБОТА №7

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СТРУКТУР

7.1. Объявление структур

Структура – это составной тип данных, в котором под одним именем объединены данные различных типов. Отдельные данные структуры называются *полями*. Объявление структуры осуществляется с помощью ключевого слова **struct**, за которым идет ее имя и далее список элементов, заключенных в фигурные скобки:

```
struct имя
{
    тип_элемента_1 имя_элемента_1;
    тип_элемента_2 имя_элемента_2;
    ...
    тип_элемента_n имя_элемента_n;
};
```

Правила работы с полями структуры идентичны работе с переменными соответствующих типов. К полям структуры можно обращаться через составное имя. Формат обращения:

имя_структуры.имя_поля

или

указатель_на_структуру->имя_поля

7.2. Пример выполнения работы

Условие. Создать массив структур, содержащий информацию о студентах: ФИО, номер группы, оценки за последнюю сессию. Вывести информацию о студентах гр. 610205 в порядке убывания среднего балла.

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

int main ()
{
    struct strc{
        char fio[40];
        char ngr[7];
        int otc[4];
        double sb;
    } mstud[100];

    int nst, i, j;
    cout << "Vvedite kol-vo studentov" << endl;
    cin >> nst;
```

```

for (i=0; i < nst; i++)
{
    cout << "Vvedite FIO";
    cin >> mstud[i].fio;

    cout << "Vvedite nomer gr.";
    cin >> mstud[i].ngr;

    cout << "Vvedite 4 otcenki" << endl;
    mstud[i].sb = 0;
for (j=0; j<4; j++)
{
    cin >> mstud[i].otc[j];
    mstud[i].sb += mstud[i].otc[j]/4.;
}
    cout << endl;
}

strc stemp;
    for (i=0; i < nst-1; i++)
for (j=i+1; j<nst; j++)
    if (mstud[i].sb < mstud[j].sb
        && !strcmp(mstud[i].ngr,"610205")
        && !strcmp(mstud[j].ngr,"610205"))
    {
        stemp = mstud[i];
        mstud[i] = mstud[j];
        mstud[j] =stemp;
    }

for (i=0; i < nst; i++)
if (!strcmp(mstud[i].ngr,"610205"))
    cout << mstud[i].fio << " " << mstud[i].ngr << " "
        << mstud[i].sb << endl;

return 0;
}

```

7.3. Индивидуальные задания

1. В магазине сформирован список постоянных клиентов, который включает ФИО, домашний адрес покупателя и размер предоставляемой скидки. Удалить из этого списка все повторяющиеся записи, проверив ФИО и домашний адрес покупателя.

2. Список товаров, имеющихся на складе, включает в себя наименование товара, количество единиц товара, цену единицы и дату поступления товара на склад. Вывести в алфавитном порядке список товаров, хранящихся больше месяца, стоимость которых превышает 100 000 р.

3. Для получения места в общежитии формируется список студентов, который включает ФИО студента, группу, средний балл, доход на члена семьи. Общежитие в первую очередь предоставляется тем, у кого доход на члена семьи меньше двух минимальных зарплат, затем остальным в порядке уменьшения среднего балла. Вывести список очередности предоставления мест в общежитии.

4. В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны его номер, тип автобуса, пункт назначения, время отправления и прибытия. Вывести информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше заданного времени.

5. На междугородной АТС информация о разговорах содержит дату разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента. Вывести для заданного города общее время разговоров с ним и сумму.

6. Информация о сотрудниках фирмы включает ФИО, табельный номер, количество отработанных часов за месяц, почасовой тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере. Вывести размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12% от суммы заработка.

7. Информация об участниках спортивных соревнований содержит наименование страны, название команды, ФИО игрока, игровой номер, возраст, рост и вес. Вывести информацию о самой молодой команде.

8. Для книг, хранящихся в библиотеке, задаются регистрационный номер книги, автор, название, год издания, издательство, количество страниц. Вывести список книг с фамилиями авторов в алфавитном порядке, изданных после заданного года.

9. Различные цехи завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают наименование, количество, номер цеха. Для заданного цеха необходимо вывести количество выпущенных изделий по каждому наименованию.

10. Информация о сотрудниках предприятия содержит ФИО, номер отдела, должность, дату начала работы. Вывести список сотрудников заданного отдела в порядке убывания стажа.

11. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит ФИО, адрес, оценки. Определить количество абитуриентов, проживающих в г. Минске и сдавших экзамены со средним баллом не ниже 4.5, вывести их фамилии в алфавитном порядке.

12. В справочной аэропорта хранится расписание вылета самолетов на следующие сутки. Для каждого рейса указаны номер рейса, тип самолета, пункт назначения, время вылета. Вывести все номера рейсов, типы самолетов и время вылета в заданный пункт назначения в порядке возрастания времени вылета.

13. У администратора железнодорожных касс хранится информация о свободных местах в поездах дальнего следования на ближайшую неделю в следующем виде: дата выезда, пункт назначения, время отправления, число свободных мест. Оргкомитет международной конференции обращается к администратору с просьбой зарезервировать m мест до города N на k -е число со временем отправления поезда не позднее t часов вечера. Вывести время отправления или сообщение о невозможности выполнить заказ в полном объеме.

14. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит ФИО абитуриента, оценки. Определить средний балл по университету и вывести список абитуриентов, средний балл которых выше среднего балла по университету. Первыми в списке должны идти студенты, сдавшие все экзамены на 10.

15. В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждая квитанция содержит следующую информацию: наименование группы изделий (телевизоры, радиоприемники и т. п.), марку изделия, дату приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Вывести информацию о состоянии заказов на текущие сутки для заданной группы изделий.

ЛАБОРАТОРНАЯ РАБОТА №8

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ

8.1. Объявление функции

Функция – это последовательность операторов, оформленная таким образом, что ее можно вызвать по имени из любого места программы. Функция описывается следующим образом:

```
тип возвращаемого значения имя функции (список параметров)  
{  
    тело функции  
}
```

Первая строка описания называется *заголовком функции*. Тип возвращаемого значения может быть любым, кроме массива или функции. Допустимо не возвращать никакого значения (тип **void**).

В C++ не допускается вложение функций друг в друга.

Выход из функции осуществляется следующими способами:

1. Если нет необходимости возвращать вычисленное значение, то выход осуществляется по достижении закрывающей скобки или при выполнении оператора **return**.

2. Если необходимо вернуть определенное значение, то выход осуществляется оператором

return выражение;

8.2. Передача параметров

При работе важно соблюдать следующее правило: при объявлении и вызове функции параметры должны соответствовать по количеству, порядку следования и типам. Функция может не иметь параметров, в этом случае после имени функции обязательно ставятся круглые скобки. Существует три основных способа передачи параметров: передача по значению, ссылке или указателю.

Передача параметров по значению

В момент обращения к функции в памяти создаются временные переменные с именами, указанными в списке параметров. В временные переменные копируются значения фактических параметров.

Передача параметров по ссылке

При передаче параметров по ссылке передается не значение соответствующей переменной, а ее адрес. Для указания на данный способ передачи после имени параметра ставится символ «&».

Передача параметров по указателю

В отличие от передачи по ссылке адрес переменной передается в функцию не с использованием операции разадресации (&), а операцией косвенной адресации (*).

8.3. Перегрузка функций и указатель на функцию

В C++ допустимо использование нескольких функций с одинаковым именем, но различным числом или типами параметров. Такое свойство называется *перегрузкой функций*. Перегруженные функции различаются компилятором по типам и числу параметров.

Так как имя функции является указателем на начало функции в оперативной памяти, то можно объявлять указатели на функции для последующего их использования в программе.

При объявлении указатель должен возвращать тот же тип и иметь такие же аргументы, как и функция, на которую он будет указывать. Например, указатель на функцию

```
double y(double x, int n);
```

будет иметь вид:

```
double (*fun)(double, int);
```

8.4. Пример выполнения работы

Условие. Вывести на экран таблицу значений функции $Y(x) = \sin x$ и ее разложения в ряд $S(x) = x - \frac{x^3}{3!} + \dots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$ с точностью $\varepsilon = 0.001$. Вывести число итераций, необходимое для достижения заданной точности.

```
#include <iostream.h>  
#include <math.h>  
#include <iomanip.h>  
typedef double (*uf)(double, double, int &);  
  
void tabl(double, double, double, double, uf);  
double y(double, double, int &);  
double s(double, double, int &);  
  
int main()  
{  
  cout << setw(8) <<"x"<< setw(15) <<"y(x)"<< setw(10) << "k" << endl;  
  tabl(0.1,0.8,0.1,0.001,y);  
  cout << endl;  
  cout << setw(8) <<"x"<< setw(15) <<"s(x)"<< setw(10) << "k" <<endl ;  
  tabl(0.1,0.8,0.1,0.001,s);  
  return 0;  
}  
  
void tabl(double a, double b, double h, double eps, uf fun)  
{  
  int k=0;
```

```

double sum;
for (double x=a; x<b+h/2; x+=h)
{
    sum=fun(x,eps,k);
    cout << setw(8) << x << setw(15) << sum << setw(10) << k << endl;
}
}

```

```

double y(double x, double eps, int &k)
{
return sin(x);
}

```

```

double s(double x, double eps, int &k)
{
double a,c,sum;
sum=a=c=x;
k=1;
while (fabs(c)>eps)
{
c = pow(x,2)/(2*k*(2*k+1));
a *= -c;
sum += a;
k++;
}
return sum;
}

```

8.5. Индивидуальные задания

Вывести на экран таблицу значений функции $Y(x)$ и ее разложения в ряд $S(x)$ с точностью ϵ (табл. 8.1). Вывести число итераций, необходимое для достижения заданной точности. Вычисление $S(x)$ и $Y(x)$ оформить в виде функций.

Таблица 8.1

№	a	b	$S(x)$	ϵ	$Y(x)$
1	2	3	4	5	6
1	-0,9	0,9	$x + \frac{x^3}{3} + \dots + \frac{x^{2k+1}}{2k+1}$	10^{-4}	$\frac{1}{2} \ln \frac{1+x}{1-x}$
2	0,1	0,9	$1 - \frac{x^2}{3!} + \dots + (-1)^k \frac{x^{2k}}{(2k+1)!}$	10^{-5}	$\frac{\sin x}{x}$
3	-0,9	0,9	$1 + \frac{x}{3} + \sum_{k=2}^{\infty} (-1)^{k-1} \frac{1 \cdot 2 \cdot 5 \cdot 8 \cdot \dots \cdot (3k-4)}{3^k k!} x^k$	10^{-3}	$\sqrt[3]{1+x}$

Окончание табл. 8.1

1	2	3	4	5	6
4	-3	3	$1 + \frac{\ln 9}{1}x + \dots + \frac{(\ln 9)^k}{k!}x^k$	10^{-4}	9^x
5	-1	1	$\frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \dots + \frac{(2k-1)!!}{(2k)!!} \cdot \frac{x^{2k+1}}{(2k+1)}$	10^{-3}	$-x + \arcsin x$
6	-0,9	0,9	$-\frac{x^2}{2 \cdot 4} + \frac{3 \cdot x^3}{2 \cdot 4 \cdot 6} + \dots + (-1)^{k-1} \frac{(2k-3)!!}{(2k)!!} x^k$	10^{-3}	$\sqrt{1+x} - 1 - \frac{x}{2}$
7	-0,5	0,5	$\frac{x^3}{3} + \frac{x^7}{7} + \dots + \frac{x^{4k-1}}{4k-1}$	10^{-5}	$\frac{1}{4} \ln \frac{1+x}{1-x} - \frac{1}{2} \operatorname{arctg} x$
8	-0,3	0,4	$\frac{1}{1+x} + \frac{2x}{1+x^2} + \dots + \frac{2^{k-1} x^{(2^{k-1}-1)}}{1+x^{(2^{k-1})}}$	10^{-4}	$\frac{1}{1-x}$
9	-2	2	$\frac{\cos x}{1} + \frac{\cos 3x}{9} + \dots + \frac{\cos(2k-1)x}{(2k-1)^2}$	10^{-4}	$\frac{\pi(\pi - 2 x)}{8}$
10	-0,5	0,5	$\sum_{k=1}^{\infty} \left(\frac{(-1)^{k+1}}{k} + \frac{(-1)^k \cdot 6}{k^3 \pi^2} \right) \sin k\pi x$	10^{-3}	$\frac{\pi x^3}{2}$
11	-1	1,3	$\sum_{k=2}^{\infty} (-1)^k \frac{\cos kx}{k^2 - 1}$	10^{-5}	$\frac{2x \sin x - 2 + \cos x}{4}$
12	1	2,5	$\sum_{k=1}^{\infty} \frac{\cos kx}{k^2}$	10^{-5}	$\frac{3x^2 - 6\pi x + 2\pi^2}{12}$
13	-1,5	1,5	$\sum_{k=1}^{\infty} (-1)^{k-1} \frac{\cos kx}{k}$	10^{-4}	$\ln(2 \cos \frac{x}{2})$
14	-0,8	0,9	$\sum_{k=2}^{\infty} \frac{(-1)^{k-1} (4k-5)!!}{(4k)!!} x^k$	10^{-4}	$\sqrt[4]{x+1} - \frac{4-x}{4}$
15	-2,5	1,3	$\sum_{k=1}^{\infty} \frac{\sin(2k-1)x}{2k-1}$	10^{-4}	$\frac{\pi \cdot \operatorname{sign} x}{4}$

ЛАБОРАТОРНАЯ РАБОТА №9

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ РЕКУРСИИ

9.1. Понятие рекурсии

Решить задачу рекурсивно – это значит разложить ее на подзадачи, которые затем аналогичным образом (т.е. рекурсивно) разбиваются на еще меньшие подзадачи. На определенном уровне подзадачи становятся настолько простыми, что могут быть решены тривиально.

В рекурсивном алгоритме важно предусмотреть способ его остановки, т.е. ввести условие, при котором рекурсивное обращение к функции прекращается.

9.2. Пример выполнения работы

Условие 1. Написать программу для вычисления $S = \sum_{i=1}^n (i+1)^2 / i$ двумя методами. Один метод вычисляет сумму без использования рекурсии, другой – с использованием рекурсии.

```
#include <iostream.h>
#include <math.h>
    double sum(int);
    double sumr(int);
int main ()
{
int n;
cout << "vvedite n "; cin >> n;
cout << "s (ne rekurs) = " << sum(n) << endl;
cout << "s (rekurs) = " << sumr(n) << endl;
return 0;
}

double sum(int n)
{
    for (double s=0, int i=1; i<=n; i++) s += (pow(i+1,2))/i;
    return s;
}

double sumr(int n)
{
if (n==1) return 4;
    else return sumr(n-1)+pow(n+1,2)/n;
}
```

Условие 2. Найти $\max(a_1, \dots, a_n)$, разбив задачу на элементарные подзадачи: $\max(\max(\max(a_1 \dots a_{n-2}), a_{n-1}), a_n), \dots$

```
int maxr2(int i)
{
    if (i==0) return a[0];
    else{
        int mx=maxr2(i-1);
        if (a[i]>mx) return a[i];
        else return mx;
    }
}
```

Условие 3. Задача о Ханойской башне. Имеется три стержня s_1, s_2, s_3 . На первом из них нанизано n дисков различных диаметров, образующих правильную пирамиду – чем выше расположен диск, тем меньше его диаметр. Требуется переместить всю башню на второй стержень, причем диски можно переносить по одному, нельзя помещать диск на диск меньшего диаметра, для промежуточного хранения можно использовать третий диск.

```
void hanr(int n, int s1, int s2, int s3)
{
    if (n>0) {
        hanr(n-1,s1,s3,s2);
        cout << "perenesty s " << s1 << " na " << s2 << endl;
        hanr(n-1,s3,s2,s1);
    }
}
```

9.3. Индивидуальные задания

Решить задачу двумя способами – с применением рекурсии и без нее.

1. Вычислить среднее значение элементов одномерного массива.
2. Вычислить произведение элементов одномерного массива.
3. Подсчитать количество цифр в заданном числе.
4. В упорядоченном массиве целых чисел $a_i, i = 1 \dots n$ найти номер элемента c методом бинарного поиска, используя очевидное соотношение: если $c \leq a_{n/2}$, тогда $c \in [a_1 \dots a_{n/2}]$, иначе $c \in [a_{n/2+1} \dots a_n]$. Если элемент c отсутствует в массиве, то вывести соответствующее сообщение.
5. Найти наибольший общий делитель чисел M и N , используя метод Эйлера: если M делится на N , то $\text{НОД}(N, M) = N$, иначе $\text{НОД}(N, M) = \text{НОД}(M \% N, N)$.
6. Вычислить значение полинома степени n по формуле

$$P_n = \sum_{i=0}^n a_i x^i = a_0 + x(a_1 + \dots x(a_{n-1} + xa_n) \dots).$$

7. Вычислить значение $x = \sqrt{a}$, используя формулу $x_n = \frac{1}{2}(x_{n-1} + a/x_{n-1})$, в качестве начального приближения использовать значение $x_0 = (1+a)/2$.

8. Найти максимальный элемент в массиве a_1, \dots, a_n , используя метод деления пополам $\max(a_1, \dots, a_n) = \max(\max(a_1, \dots, a_{n/2}), \max(a_{n/2+1}, \dots, a_n))$.

9. Вычислить $y(n) = \sqrt{1 + \sqrt{2 + \dots + \sqrt{n}}}$.

10. Вычислить произведение $n \geq 2$ (n четное) сомножителей

$$y = \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \dots$$

11. Вычислить $y = x^N$ по следующему алгоритму: $y = (x^{N/2})^2$, если N четное; $y = x \cdot x^{N-1}$, если N нечетное.

12. Вычислить $y(n) = \frac{1}{n + \frac{1}{(n-1) + \frac{1}{(n-2) + \frac{1}{\dots + \frac{1}{1 + \frac{1}{2}}}}}}$

13. Вычислить произведение двух целых положительных чисел $p = a \cdot b$ по следующему алгоритму: $p = 2 \cdot (a \cdot b / 2)$, если b четное; $p = a + (a \cdot (b - 1))$, если b нечетное. Если $b = 0$, то $p = 0$.

14. Вычислить значение суммы $S = 1/1! + 1/2! + \dots + 1/k!$

15. Проверить, является ли заданная строка палиндромом.

ЛАБОРАТОРНАЯ РАБОТА №10

ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ФАЙЛОВ

10.1. Организация работы с файлами

Различают два вида файлов: текстовые и двоичные.

Текстовые файлы хранят информацию в виде последовательности символов. В текстовом режиме каждый разделительный символ строки автоматически преобразуется в пару (возврат каретки – переход на новую строку).

Бинарные (или двоичные) файлы предназначены для хранения только числовых значений данных. Структура такого файла определяется программно.

Функции для работы с файлами размещены в библиотеках `stdio.lib` (`#include <stdio.h>`) и `io.lib` (`#include <io.h>`). Каждый файл должен быть связан с некоторым указателем. Этот указатель имеет тип `FILE` и используется во всех операциях с файлами.

Формат объявления указателя на файл следующий:

FILE *указатель на файл;

Макрос **NULL** определяет пустой указатель.

Макрос **EOF**, часто определяемый как `-1`, является значением, возвращаемым тогда, когда функция ввода пытается выполнить чтение после конца файла.

Макрос **FOPEN_MAX** определяет целое значение, равное максимальному числу одновременно открытых файлов.

10.2. Функции для работы с файлами

Функция

**FILE *fopen(const char *имя_файла,
const char *режим_открытия);**

открывает файл и связывает его с потоком. Возвращает указатель на открытый файл. *Имя_файла* – это указатель на строку символов, в которой хранится имя файла и путь к нему. *Режим_открытия* – это указатель на строку символов, в которой указывается режим открытия файла. Допустимые режимы:

r – открытие текстового файла для чтения;

w – создание текстового файла для записи;

a – добавление информации в конец текстового файла.

При работе с текстовыми файлами к символу, указывающему режим открытия, добавляется символ «*t*» (по умолчанию), а при работе с бинарными – «*b*». Если необходимо и читать и записывать в файл, то добавляется символ «*+*». При возникновении ошибки во время открытия файла, функция **fopen** возвращает значение **NULL**.

Функция

int fclose(FILE *указатель_на_файл);

закрывает поток, который был открыт с помощью вызова **fopen()** и записывает в файл все данные, которые еще оставались в дисковом буфере. Доступ к файлу после выполнения функции будет запрещен.

Возвращение нуля означает успешную операцию закрытия. В случае же ошибки возвращается EOF.

Функция

int fcloseall(void);

закрывает все открытые файлы. Возвращает количество закрытых файлов или EOF, если возникает ошибка.

Функция

int putc(int символ, FILE * указатель_на_файл);

записывает один символ в текущую позицию указанного открытого файла.

Функция

int getc(FILE * указатель_на_файл);

читает один символ из текущей позиции указанного открытого файла.

Функция

int feof(FILE * указатель_на_файл);

возвращает отличное от нуля значение (true), если конец файла не достигнут, и ноль (false), если достигнут конец файла.

Функция

int fputs(const char * строка, FILE * указатель_на_файл);

записывает строку символов в текущую позицию указанного открытого файла.

Функция

char * fgets(char * строка, int длина, FILE * указатель_на_файл);

читает строку символов из текущей позиции указанного открытого файла до тех пор, пока не будет прочитан символ перехода на новую строку, или количество прочитанных символов не станет равным (*длина* – 1).

Функция

int * fprintf(FILE * указатель_на_файл, const char * управляющая_строка);

записывает форматированные данные в файл. *Управляющая_строка* определяет строку форматирования аргументов, заданных своими адресами. Обычно эта строка состоит из последовательности символов «%», после которых следует символ типа данных:

I или *i* – десятичное, восьмеричное или шестнадцатеричное целое;

D или *d* – десятичное целое;

U или *u* – десятичное целое без знака;

E или *e* – действительное с плавающей точкой;

s – строка символов;

c – символ.

Функция

int * fscanf(FILE * указатель_на_файл, const char * управляющая_строка);

читает форматированные данные из файла. Строка форматирования строится аналогично функции **fprintf**.

Функция

```
void rewind(FILE * указатель_на_файл);
```

устанавливает указатель текущей позиции выделенного файла в начало файла.

Функция

```
int ferror(FILE * указатель_на_файл);
```

определяет, произошла ли ошибка во время работы с файлом.

Функция

```
size_t fwrite(const void * записываемое_данные,  
size_t размер_элемента, size_t число_элементов,  
FILE * указатель_на_файл);
```

записывает в файл заданное число данных определенного размера. Размер данных задается в байтах. Тип `size_t` определяется как целое без знака.

Функция

```
size_t fread(void * считываемое_данные,  
size_t размер_элемента, size_t число_элементов,  
FILE * указатель_на_файл);
```

считывает из файла указанное число данных заданного размера. Размер задается в байтах. Функция возвращает число прочитанных элементов. Если число прочитанных элементов не равно заданному, то при чтении возникла ошибка или встретился конец файла.

Функция

```
int fileno(FILE * указатель_на_файл);
```

возвращает значение дескриптора указанного файла (дескриптор – логический номер файла для заданного потока).

Функция

```
long filelength(int дескриптор);
```

возвращает длину файла с соответствующим дескриптором в байтах.

Функция

```
int fseek(FILE * указатель_на_файл, long int число_байт,  
int точка_отсчета);
```

устанавливает указатель в заданную позицию. Заданное количество байт отсчитывается от позиции, которая задается следующими макросами: `SEEK_SET` – начало файла, `SEEK_CUR` – текущая позиция, `SEEK_END` – конец файла.

10.3. Пример выполнения работы

Условие. Написать программу, вводящую в файл или читающую из файла ведомость студентов, сдавших экзамены. Каждая структура должна содержать фамилию, а также оценки по математике и программированию. Вывести список студентов, сдавших экзамен по программированию с оценкой 4, и записать эту информацию в текстовый файл.

```
#include <iostream.h>  
#include <stdio.h>  
#include <conio.h>
```

```

#include <stdlib.h>
#include <string.h>

FILE *f;
typedef struct
{
    char fio[30];
    unsigned char matem;
    unsigned char oaip;
} TStudent;

TStudent stud[30]; // Массив структур
char name[20];    // Имя файла
int nst=0;        // Число введенных структур
int menu();      // Меню
void nnf();      // Ввести имя файла
void newf();     // Создать новый файл
void spisok();  // Ввести список
void opf();      // Открыть файл
void resc();    // Вывести результат на экран
void resf();    // Вывести результат в файл

int main()
{
    while (true)
    {
        switch (menu())
        {
            case 1: nnf(); break;
            case 2: newf(); break;
            case 3: spisok(); break;
            case 4: opf(); break;
            case 5: resc(); break;
            case 6: resf(); break;
            case 7: return 0;
            default: "Viberite pravilno!";
        }
        puts("Press any key to continue");
        getch(); system("cls");
    }
}

int menu() // Меню
{

```

```

cout << "VIBERITE:" << endl;
cout << "1. Vvod file name" << endl;
cout << "2. New file" << endl;
cout << "3. Vvesti spisok" << endl;
cout << "4. Open file" << endl;
cout << "5. Vivesti result" << endl;
cout << "6. Vivesti v fail" << endl;
cout << "7. Exit" << endl;
int i;
cin >> i;
return i;
}

void nnf()      // Ввести имя файла
{
    cout << "Vvedite file name" << endl;
    cin >> name;
}

void newf()    // Создать новый файл
{
    if ((fl = fopen(name,"wb"))==NULL)
    {
        cout << "Oshibka pri sozdanii"<<endl;
        exit(1);
    }
    cout << "OK" << endl;
    fclose(fl);
}

void spisok()  // Ввести список
{
    if ((fl = fopen(name,"rb+"))==NULL)
    {
        cout << "Oshibka pri sozdanii"<<endl;
        exit(1);
    }

    cout << "Vvedite chislo studentov " << endl;
    cin >> nst;

    for (int i=0; i<nst; i++)
    {
        cout << "Vvedite imya: ";

```

```

    cin >> stud[i].fio;
    cout << "Vvedite otcenku po matematike: ";
    cin >> stud[i].matem;
    cout << "Vvedite otcenku po OAiP: ";
    cin >> stud[i].oaip;
    fwrite( &stud[i], sizeof(TStudent), 1, fl );
}
fclose(fl);
}

void opf()      // Открыть файл
{
if ((fl = fopen(name,"rb"))==NULL)
{
    cout << "Oshibka pri otkritii"<<endl;
    exit(1);
}

    nst=0; TStudent std;
while(true)
{
int nwrt = fread( &std, sizeof(TStudent), 1, fl );
if (nwrt!=1) break;
    stud[nst]=std;
    cout << stud[nst].fio << " " << stud[nst].matem
        << " " << stud[nst].oaip << endl;

    nst++;
}
fclose(fl);
}

void resc()    // Вывести результат на экран
{
for (int i=0; i<nst; i++)
if (stud[i].oaip=='4')
    cout << stud[i].fio << endl;
}

void resf()    // Вывести результат в файл
{
char namet[30];
FILE *ft;
    cout << "Vvedite imya faila" << endl;
    cin >> namet;

```

```

if ((ft = fopen(namet,"w"))==NULL)
{
    cout << "Oshibka pri sozdanii "<<endl;
    exit(1);
}
char s[80];
for (int i=0; i<nst; i++)
if (stud[i].oaip=='4')
{
    strcpy(s, stud[i].fio);
    strcat(s, "\n");           // Добавление разделителя строк
    fputs(s, ft);
}
fclose(ft);
}

```

10.4. Индивидуальные задания

В программе предусмотреть сохранение вводимых данных в файл и возможность чтения из ранее сохраненного файла. Результаты выводить на экран и в текстовой файл.

Внимание! Разработанная программа будет использоваться в других лабораторных работах.

1. Список товаров, имеющихся на складе, включает в себя наименование товара, количество единиц товара, цену единицы и дату поступления товара на склад. Вывести список товаров, хранящихся больше месяца, стоимость которых превышает 1 000 000 р.

2. Для получения места в общежитии формируется список студентов, который включает ФИО студента, группу, средний балл, доход на члена семьи. Вывести информацию о студентах, у которых доход на члена семьи менее двух минимальных зарплат.

3. В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны его номер, пункт назначения, время отправления и прибытия. Вывести информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше заданного времени.

4. Информация о сотрудниках фирмы включает ФИО, количество отработанных часов за месяц, почасовой тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере. Вывести размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12 % от суммы заработка.

5. Информация об участниках спортивных соревнований содержит название команды, ФИО игрока, возраст. Вывести информацию о спортсменах, возраст которых не достиг 18 лет.

6. Для книг, хранящихся в библиотеке, задаются автор, название, год издания, количество страниц. Вывести список книг, изданных после заданного года.

7. На заводе выпускается несколько наименований деталей. Сведения о деталях включают код детали, количество выпущенных деталей, номер месяца выпуска. Вывести информацию о продукции, выпущенной заданным цехом за последний месяц.

8. Информация о сотрудниках предприятия содержит ФИО, номер отдела, должность, дату начала работы. Вывести список сотрудников заданного отдела, проработавших на предприятии более 20 лет.

9. Ведомость абитуриентов содержит ФИО, город проживания, суммарный балл. Вывести информацию об абитуриентах, проживающих в г. Минске и имеющих балл больше 220.

10. В справочной аэропорта хранится расписание вылета самолетов на следующие сутки. Для каждого рейса указаны номер рейса, пункт назначения, время вылета. Вывести все номера рейсов и время вылета самолета для заданного пункта назначения.

11. У администратора железнодорожных касс хранится информация о свободных местах в поездах. Информация представлена в следующем виде: номер поезда, пункт назначения, время отправления, число свободных мест. Вывести информацию о поездах, в которых имеются свободные места, до заданного пункта назначения.

12. Ведомость студентов, сдававших сессию, содержит ФИО и оценки по четырем предметам. Вывести список студентов, сдавших сессию со средним баллом больше 7.

13. В радиоателье хранятся квитанции о сданных в ремонт телевизорах. Каждая квитанция содержит следующую информацию: марка телевизора, дата приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Вывести информацию о заказах, которые на текущий момент не выполнены.

14. На АТС информация о разговорах содержит номер телефона абонента, время разговора и тариф. Вывести для заданного абонента сумму, которую ему следует оплатить за разговоры.

15. В магазине составлен список людей, которым выдана карта постоянного покупателя. Каждая запись этого списка содержит номер карточки, ФИО, предоставляемую скидку. Вывести информацию о покупателях, имеющих 10 %-ную скидку в магазине.

ЛАБОРАТОРНАЯ РАБОТА №11

СОРТИРОВКА ПО КЛЮЧУ ОДНОМЕРНЫХ МАССИВОВ СТРУКТУР

11.1. Сортировка массивов

Метод пузырька

```
void s_puz(int a[], int n)
{
    int i,j,t;
    for(i=1; i < n; i++)
        for( j=n-1; j >= i; j--)
            if (a[j-1] > a[j])
                { t = a[j-1]; a[j-1] = a[j]; a[j] = t; }
}
```

Сортировка выбором

```
void s_vb(int a[], int n)
{
    int imin,i,j,t;

    for(i=0; i<n-1; i++)
    {
        imin=i;
        for(j=i+1; j<n; j++)
            if (a[imin]>a[j]) imin=j;
        if (imin != i)
        {
            t = a[imin];
            a[imin] = a[i];
            a[i] = t;
        }
    }
}
```

Сортировка вставками

```
void s_vst(int a[], int n)
{
    int i,j,t;
    for(i=1; i<n; i++)
    {
```

```

        t=a[i];
        for(j=i-1; j>=0 && t<a[j]; j--) a[j+1]=a[j];
            a[j+1] = t;
    }
}

```

11.2. Пример выполнения работы

Функция для сортировки методом QuickSort. Сортировка ведется по ключу – оценка по ОАиП (используются типы данных, объявленные в примере пункта 10.3.

```

void s_qs(TStudent st[], int n)
{

struct
{
    int l;
    int r;
} stack[20];

int i,j, left, right, x, s=0;
    TStudent t;

    stack[s].l=0; stack[s].r=n-1;
while (s != -1)
    {
        left=stack[s].l; right=stack[s].r;
        s--;
while (left < right)
        {
            i=left; j=right; x=st[(left+right)/2].oaip;
while (i <= j)
            {
while (st[i].oaip < x) i++;
while (st[j].oaip > x) j--;
if (i<=j) {
                t=st[i]; st[i]=st[j]; st[j]=t;
                i++; j--;
            }
        }
if ((j-left)<(right-i))
        {
if (i<right) {s++; stack[s].l=i; stack[s].r=right; }
            right=j;
        }
    }
}

```


ЛАБОРАТОРНАЯ РАБОТА №12 ПОИСК ПО КЛЮЧУ В ОДНОМЕРНОМ МАССИВЕ СТРУКТУР

12.1. Поиск в массиве

Линейный поиск (метод полного перебора)

```
int p_lin1(int a[],int n, int x)
{
for(int i=0; i < n; i++)
    if (a[i]==x) return i;
return -1;
}
```

Двоичный поиск

```
int p_dv(int a[],int n, int x)
{
int i=0, j=n-1, m;
while(i<j)
{
m=(i+j)/2;
if (x > a[m]) i=m+1; else j=m;
}
if (a[i]==x) return i;
else return -1;
}
```

Интерполяционный поиск. Обычно 1 – 2 первых шага делается с помощью интерполяционного поиска, а затем используется двоичный поиск.

```
int p_dv(int a[],int n, int x)
{
int i=0, j=n-1, m;
while(i<j)
{
if (a[i]==a[j]) //Предотвращение деления на нуль
    if (a[i]==x) return i;
    else return -1;

m=i+(j-i)*(x-a[i])/(a[j]-a[i]);

if (a[m]==x) return m;
}
```

```

else
if (x > a[m]) i=m+1; else j=m-1;
}

return -1;
}

```

12.2. Индивидуальные задания

Использовать программу, составленную при выполнении лабораторной работы №11. Найти в отсортированном массиве структур заданный элемент указанными методами поиска (для упрощения предполагаем, что в массиве присутствует только один элемент с нужными характеристиками).

1. Найти товар ценой 150 000 р. Методы поиска: полный перебор и двоичный.
2. Найти студента, имеющего средний балл 7,3. Методы поиска: полный перебор и интерполяционный.
3. Найти автобус, отправляющийся в рейс в 13.00. Методы поиска: полный перебор и двоичный.
4. Найти сотрудника, отработавшего за месяц 156 часов. Методы поиска: полный перебор и интерполяционный.
5. Найти спортсмена, которому 28 лет. Методы поиска: полный перебор и двоичный.
6. Найти книгу 1966 г. издания. Методы поиска: полный перебор и интерполяционный.
7. Найти деталь с кодом 09383. Методы поиска: полный перебор и двоичный.
8. Найти сотрудника, работающего с 1975 г. Методы поиска: полный перебор и интерполяционный.
9. Найти абитуриента, набравшего 287 баллов. Методы поиска: полный перебор и двоичный.
10. Найти самолет, вылетающий в 14.00. Методы поиска: полный перебор и интерполяционный.
11. Найти поезд, отправляющийся в 21.00. Методы поиска: полный перебор и двоичный.
12. Найти студента со средним баллом 8,3. Методы поиска: полный перебор и интерполяционный.
13. Найти телевизор, сданный в ремонт 25 числа. Методы поиска: полный перебор и двоичный.
14. Найти абонента с номером 21603. Методы поиска: полный перебор и интерполяционный.
15. Найти покупателя с номером карточки 00458. Методы поиска: полный перебор и двоичный.

ЛАБОРАТОРНАЯ РАБОТА №13 ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ОДНОНАПРАВЛЕННЫХ СПИСКОВ ТИПА «СТЕК»

13.1. Работа со стеками

Объявлена структура следующего типа:

```
struct tstk  
{ int inf;  
  tstk *a; } sp;
```

Добавление элемента в стек

```
tstk *AddStask(tstk *sp, int inf)  
{ tstk *spt=new tstk;  
  spt->inf = inf;  
  spt->a = sp;  
  return spt; }
```

Чтение элемента с удалением

```
tstk *ReadStackD(tstk *sp, int &inf)  
{ if (sp == NULL) return NULL;  
  tstk *spt = sp;  
  inf= sp->inf;  
  sp = sp->a;  
  delete spt;  
  return sp; }
```

Удаление всего стека

```
tstk *DelStackAll(tstk *sp)  
{ tstk *spt; int inf;  
  while(sp != NULL) {  
    spt = sp;  
    inf= sp->inf;  
    cout << inf << endl;  
    sp = sp->a;  
    delete spt; }  
  return NULL; }
```

Обмен следующих за текущим элементов

```
void RevStackAfter(tstk *sp)
{ tstk *spt = sp->a->a;
  sp->a->a = spt->a;
  spt->a = sp->a;
  sp->a = spt; }
```

13.2. Индивидуальные задания

Создать стек с числами в диапазоне от –50 до +50. После создания стека выполнить индивидуальное задание. В конце работы все стеки должны быть удалены.

1. Преобразовать стек в два стека. Первый должен содержать только положительные числа, а второй – отрицательные.
2. Создать новый стек, содержащий только четные числа из первого стека.
3. Создать новый стек, содержащий только те числа из первого стека, которые больше среднего значения всех элементов первого стека.
4. Создать новый стек, содержащий только положительные числа из первого стека.
5. Подсчитать, сколько элементов стека имеет значение, которое превышает среднее значение всех элементов стека.
6. Найти максимальное и минимальное значения стека.
7. Определить, сколько элементов стека, начиная с вершины, находится до элемента с максимальным значением.
8. Определить, сколько элементов стека, начиная от вершины, находится до элемента с минимальным значением.
9. Определить, сколько элементов стека находится между его минимальным и максимальным элементами.
10. Определить, сколько элементов стека имеет значение меньше среднего значения всех элементов стека.
11. Создать новый стек, содержащий только числа, большие среднего значения всех элементов первого стека.
12. Преобразовать стек в два стека. В первый поместить все четные, а во второй – все нечетные числа.
13. Создать новый стек, порядок следования элементов в котором обратный относительно первого стека.
14. Создать новый стек, в который поместить каждый третий элемент первого стека.
15. Создать новый стек, в который поместить элементы, лежащие во второй половине первого стека.

ЛАБОРАТОРНАЯ РАБОТА №14 ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ОДНОНАПРАВЛЕННЫХ СПИСКОВ ТИПА «ОЧЕРЕДЬ»

14.1. Работа с однонаправленными списками

Очередь – это упорядоченный список, в котором элементы добавляются в один конец списка, а извлекаются из другого конца. Для простой очереди можно использовать методы, аналогичные методам работы со стеками.

Добавление элемента в очередь

```
void AddOch(toch **sp,toch **spt, int inf)
{
    toch *spt=new toch;
    spt->inf = inf;
    spt->a = NULL;
    if (*sp == NULL) // Если нет элементов
        *sp=*spt=spt;
    else
    { (*sp)->a = spt;  *sp = spt; }
    return;
}
```

Подключение:

```
sp=spt=NULL;
AddOch(&sp, &spt, информация);
```

Чтение элемента с удалением

```
toch *ReadOchD(toch *sp, int &inf)
{
    if (sp == NULL) return NULL;
    inf= sp->inf;
    toch *spt = sp;
    sp = sp->a;
    delete spt;
    return sp;
}
```

Удаление элемента, следующего за текущим

```
void DelOchAfter(toch *sp)
{
    if (sp->a == NULL) return;
    toch *spt = sp->a;
    sp->a = spt->a->a;
}
```

```
    delete spt;
}
```

Удаление всей очереди

```
void DelOchAll(toch **sp, toch **spk)
{
    toch *spt; int inf;
    while(*sp != NULL)
    {
        spt = *sp;
        inf= (*sp)->inf;
        cout << inf << endl;
        *sp = (*sp)->a;
        delete spt;
    }
    *spk=NULL;
}
```

14.2. Индивидуальные задания

Создать однонаправленную очередь с числами в диапазоне от –50 до +50. После создания очереди выполнить индивидуальное задание. В конце работы все очереди должны быть удалены.

1. Удалить из очереди все четные числа.
2. Удалить из очереди все отрицательные числа.
3. Поменять местами крайние элементы очереди.
4. Поменять местами минимальный и максимальный элементы очереди.
5. Удалить из очереди каждый второй элемент.
6. Удалить из очереди все элементы, расположенные до минимального элемента очереди.
7. Поменять местами наибольший среди отрицательных и наименьший среди положительных элементов очереди.
8. Поместить максимальный элемент очереди на первую позицию.
9. Поменять местами минимальный и первый элементы очереди.
10. Поменять местами первый и последний элементы очереди.
11. Удалить первый и последний элементы очереди.
12. Удалить из очереди все элементы, расположенные между минимальным и максимальным элементами очереди.
13. Удалить из очереди все элементы, стоящие после максимального элемента.
14. Найти среднее значение всех элементов очереди и удалить все элементы, которые меньше среднего значения.
15. Найти среднее значение всех элементов очереди. Поместить ближайший к среднему значению элемент очереди на первую позицию.

ЛАБОРАТОРНАЯ РАБОТА №15 ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ДВУСВЯЗАННЫХ СПИСКОВ

15.1. Очереди на основе двусвязанных списков

Двусвязанный список состоит из элементов, содержащих ссылки как на предыдущий, так и на последующий элементы. Такая организация позволяет осуществлять перемещение по списку в любом направлении. Это упрощает работу со списком, в частности, вставку и удаление. Также повышается устойчивость работы, т.к. можно восстановить потерянную ссылку. Недостатком является введение дополнительного указателя, что несколько усложняет программу.

Объявление рекурсивного типа – двусвязанная очередь:

```
struct tochd
{
    int inf;
    tochd *left;
    tochd *righth;
} *sp;
```

Создание очереди

```
void NewOchd(tochd **sl, tochd **sr)
{
    *sl=new tochd;
    *sr=new tochd;
    (*sl)->left = NULL;
    (*sl)->righth = *sr;
    (*sr)->left = *sl;
    (*sr)->righth = NULL;
    return;
}
```

Подключение

```
tochd *sl, *sr;
NewOchd(&sl,&sr);
```

Добавление элемента после заданного

```
void AddOchdRighth(tochd *sp, int inf)
{
    tochd *spt=new tochd;
    spt->inf = inf;
```

```

    spt->left = sp;
    spt->right = sp->right;
    sp->right = spt;
    spt->right->left = spt;
    return;
}

```

Добавление элемента перед заданным

```

void AddOchdLeft(tochd *sp, int inf)
{
    tochd *spt=new tochd;
    spt->inf = inf;
    spt->left = sp->left;
    spt->right = sp;
    spt->left->right = spt;
    sp->left = spt;
    return;
}

```

Чтение и удаление элемента с адресом sp

```

int ReadOchdD(tochd *sp)
{
    int inf= sp->inf;
    sp->left->right = sp->right;
    sp->right->left = sp->left;
    delete sp;
    return inf;
}

```

Удаление всей очереди

```

void DelOchdAll(tochd **sl, tochd **sr)
{
    tochd *spt = (*sl)->right;
    while(spt != *sr)
    {
        cout << ReadOchdD(spt) << endl;
        spt = (*sl)->right;
    }
    delete *sl; *sl = NULL;
    delete *sr; *sr = NULL;
    return;
}

```

Сортировка слиянием

Разбиение списка на 2 списка

```
void div2Ochd(tochd *sl, tochd *sr, tochd **sL,
              tochd **srL, tochd **sR, tochd **srR)
{
  NewOchd(sL, srL);
  NewOchd(sR, srR);
  tochd *spt = sl->right;
  while(spt != sr)
  {
    AddOchdLeft(*srL, ReadOchdD(spt));
    spt = sl->right;
    if (spt != sr)
    {
      AddOchdLeft(*srR, ReadOchdD(spt));
      spt = sl->right;
    }
  }
  delete sl;
  delete sr;
}
```

Слияние двух отсортированных списков

```
void slipOchd(tochd **sl, tochd **sr, tochd *sL, tochd *srL, tochd
*sR, tochd *srR)
{
  NewOchd(sl, sr);
  tochd *sptL = sL->right;
  tochd *sptR = sR->right;
  while ((sptL != srL) && (sptR != srR))
  {
    if (sptL->inf < sptR->inf)
    {
      AddOchdLeft(*sr, ReadOchdD(sptL));
      sptL = sL->right;
    }
    else
    {
      AddOchdLeft(*sr, ReadOchdD(sptR));
      sptR = sR->right;
    }
  }

  while (sptL != srL)
```

```

    {
        AddOchdLeft(*sr, ReadOchdD(sptL));
        sptL = slL->right;
    }
    delete slL; delete srL;

    while (sptR != srR)
    {
        AddOchdLeft(*sr, ReadOchdD(sptR));
        sptR = slR->right;
    }
    delete slR; delete srR;
}

```

Сортировка

```

void SotrSlipOchd(tochd **sl, tochd **sr)
{
    tochd *slL, *srL, *slR, *srR;
    if ((*sl)->right->right == *sr) return;

    div2Ochd(*sl, *sr, &slL, &srL, &slR, &srR);
    SotrSlipOchd(&slL, &srL);
    SotrSlipOchd(&slR, &srR);

    slipOchd(sl, sr, slL, srL, slR, srR);
}

```

15.2. Индивидуальные задания

Создать двунаправленный список с числами в диапазоне от –50 до +50. После создания списка выполнить индивидуальное задание и вывести результат. Написать программу сортировки и поиска в двунаправленном списке. Отсортировать список и вывести его на экран. Найти элемент списка, равный номеру варианта, и вывести его порядковый номер либо сообщение о том, что такого элемента нет. В конце работы все списки должны быть удалены.

1. Найти минимальный элемент и сделать его первым.
2. Создать два списка. Первый должен содержать только положительные, а второй – только отрицательные числа.
3. Удалить из списка все элементы, находящиеся между его максимальным и минимальным значениями.
4. Переместить во второй список все элементы, находящиеся после элемента с максимальным значением.
5. Сместить по кольцу элементы списка на заданное число позиций.
6. Удалить все отрицательные элементы списка.

7. Из элементов, расположенных между максимальным и минимальным элементами, создать кольцо. Остальные элементы должны остаться в первом списке.

8. Удалить первый и последний элементы списка.

9. Удалить по одному элементу справа и слева от минимального элемента. Если элементы справа или слева отсутствуют, то удалить сам минимальный элемент.

10. Удалить элементы, заключенные между максимальным и минимальным значениями.

11. Удалить из списка элементы с повторяющимися более одного раза значениями.

12. Поменять местами элементы с максимальным и минимальным значениями, при этом элементы не должны перемещаться в памяти.

13. Преобразовать список в кольцо. Предусмотреть возможность движения по кольцу в обе стороны с отображением места положения текущего элемента. В конце работы восстановить начальный список.

14. Поменять местами первый и максимальный элементы списка.

15. Поменять местами первый и последний элементы списка.

ЛАБОРАТОРНАЯ РАБОТА №16 ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ДРЕВОВИДНЫХ СТРУКТУР ДАННЫХ

16.1. Основные операции с бинарным деревом поиска

Объявлена структура следующего типа:

```
struct ttree
{
    int inf;
    ttree *left;
    ttree *righth;
} *proot;
```

Добавление нового элемента

```
ttree *addtree(ttree *proot, int inf)
{
    ttree *nl, *pr, *ps;
    bool b;
    nl = new ttree;
    nl->inf = inf;
    nl->left = NULL;
    nl->righth = NULL;
    if (proot == NULL) return nl;
    ps = proot;
    while (ps != NULL)
    {
        pr=ps;
        b = (inf < ps->inf);
        if (b) ps = ps->left;
        else ps = ps->righth;
    }
    if (b) pr->left = nl;
    else pr->righth = nl;
    return proot;
}
```

Обход всего дерева

```
void wrtree(ttree *p)
{
    if (p==NULL) return;
```

```

wrtree(p->left);
cout << p->inf << " ";
wrtree(p->right);
}

```

Поиск элемента с заданным ключом

```

void poisktree(ttree *p,int key, bool &b, int &inf)
{
if ((p != NULL) && (b != true))
{
    if (p->inf !=key)
    {
        poisktree(p->left, key,b,inf);
        poisktree(p->right,key,b,inf);
    }
    else {
        b=true;
        inf=p->inf;
    }
}
return;
}

```

Поиск элемента с максимальным ключом

```

int poiskmaxtree(ttree *p)
{
while (p->right != NULL) p = p->right;
return p->inf;
}

```

Удаление всего дерева

```

ttree *deltree(ttree *p)
{
if (p==NULL) return NULL;
deltree(p->left);
deltree(p->right);
delete(p);
p = NULL;
return NULL;
}

```

Удаление элемента с заданным ключом

```
ttree *dellist(ttree *proot, int inf)
{
    ttree *ps = proot, *pr = proot, *w, *v;

    // Поиск удаляемого узла
    while ((ps != NULL) && (ps->inf != inf))
    {
        pr = ps;
        if (inf < ps->inf) ps = ps->left;
        else ps = ps->right;
    }

    if (ps == NULL) return proot; // Если узел не найден

    // Если узел не имеет дочерей
    if ((ps->left == NULL) && (ps->right == NULL))
    {
        if (ps == pr) // Если это был последний элемент
        {
            delete(ps);
            return NULL;
        }
        if (pr->left == ps) // Если удаляемый узел слева
            pr->left = NULL;
        else // Если удаляемый узел справа
            pr->right = NULL;
        delete(ps);
        return proot;
    }

    // Если узел имеет дочь только справа
    if (ps->left == NULL)
    {
        if (ps == pr) // Если удаляется корень
        {
            ps = ps->right;
            delete(pr);
            return ps;
        }
    }

    if (pr->left == ps) // Если удаляемый узел слева
        pr->left = ps->right;
```

```

else // Если удаляемый узел справа
    pr->right = ps->right;
delete(ps);
return proot;
}

// Если узел имеет дочь только слева
if (ps->right == NULL)
{
if (ps == pr) // Если удаляется корень
    {
    ps = ps->left;
delete(pr);
return ps;
    }

if (pr->left == ps) // Если удаляемый узел слева
    pr->left = ps->left;
else // Если удаляемый узел справа
    pr->right = ps->left;
delete(ps);
return proot;
}

// Если узел имеет двух дочерей
w = ps->left;
if (w->right == NULL) // Если максимальный следует за ps
    w->right = ps->right;
else // Если максимальный не следует за ps
    {
    while (w->right != NULL)
    {
    v = w;
    w = w->right;
    }
v->right = w->left;
w->left = ps->left;
w->right = ps->right;
    }

if (ps == pr) // Если удаляется корень
    {
    delete(ps);
return w;
    }

```

```

    }
    if (pr->left == ps) // Если удаляемый узел слева
        pr->left = w;
    else // Если удаляемый узел справа
        pr->right = w;
    delete(ps);
    return proot;
}

```

16.2. Индивидуальные задания

Создать сбалансированное дерево поиска с числами в диапазоне от –50 до +50 и распечатать информацию прямым, обратным обходом и в порядке возрастания. Написать функции добавления нового значения, удаления значения, поиска значения. Выполнить индивидуальное задание и вывести результат. Вся выделенная память должна быть освобождена.

1. Поменять местами информацию, содержащую максимальный и минимальный ключи.
2. Подсчитать число листьев в дереве (Лист – это узел, из которого нет ссылок на другие узлы дерева).
3. Удалить из дерева ветвь с вершиной, имеющей заданный ключ.
4. Определить максимальную глубину дерева, т.е. число узлов в самом длинном пути от корня дерева до листьев.
5. Определить число узлов на каждом уровне дерева.
6. Удалить из левой ветви дерева узел с максимальным значением ключа и все связанные с ним узлы.
7. Определить количество символов во всех строках, находящихся в узлах дерева.
8. Определить число листьев на каждом уровне дерева.
9. Определить число узлов в дереве, в которых есть указатель только на одну ветвь.
10. Определить число узлов в дереве, у которых есть две дочери.
11. Определить количество записей в дереве, начинающихся с определенной буквы (например «а»).
12. Найти среднее значение всех ключей дерева и узел, имеющий ближайший к этому значению ключ.
13. Найти запись с ключом, ближайшим к среднему значению между максимальным и минимальным значениями ключей.
14. Определить количество узлов в левой ветви дерева.
15. Определить количество узлов в правой ветви дерева.

ЛИТЕРАТУРА

1. Основы алгоритмизации и программирования. Язык Си : учеб. пособие / М. П. Батура [и др.] – Минск : БГУИР, 2007.
2. Основы алгоритмизации и программирования : конспект лекций для студ. всех спец. и всех форм обуч. БГУИР / В. Л. Бусько [и др.] – Минск : БГУИР, 2004.
3. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. – СПб. : Невский диалект, 2005.
4. Кнут, Д. Искусство программирования. В 3 т. Т 3 : Сортировка и поиск / Д. Кнут. – М. : Вильямс, 2000.
5. Хопкрофт, Дж. Структуры данных и алгоритмы / Дж. Хопкрофт, Дж. Ульман, А. Ахо. – М. : Вильямс, 2003.
6. Павловская, Т. А. С/С++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.
7. Павловская, Т. А. С++. Объектно-ориентированное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2004.
8. Керниган, Б. Язык программирования Си / Б. Керниган, Д. Ритчи. – М. : Финансы и статистика, 1992.
9. Демидович, Е. М. Основы алгоритмизации и программирования. Язык СИ / Е. М. Демидович. – Минск : Бестпринт, 2001.
10. Страуструп, Б. Язык программирования С++ / Б. Страуструп. – СПб. : БИНОМ, 1999.

Св. план 2008, резерв

Учебное издание

Бусько Виталий Леонидович
Навроцкий Анатолий Александрович

**ОСНОВЫ АЛГОРИТМИЗАЦИИ
И ПРОГРАММИРОВАНИЯ В СРЕДЕ VISUAL C++**

Лабораторный практикум по курсу
«Основы алгоритмизации и программирования»
для студентов 1 – 2-го курсов всех специальностей БГУИР

Редактор Е. Н. Батурчик
Корректор Л. А. Шичко

Подписано в печать 11.12.2008.
Гарнитура «Таймс».
Уч.-изд. л. 4,0.

Формат 60×84 1/16.
Печать ризографическая.
Тираж 500 экз.

Бумага офсетная.
Усл. печ. л. 4,07.
Заказ 378.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0056964 от 01.04.2004. ЛП №02330/0131666 от 30.04.2004.
220013, Минск, П. Бровки, 6