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

Кафедра вычислительных методов и программирования

А. А. Навроцкий, И. Н. Коренская, В. Л. Бусько

ОСНОВЫ АЛГОРИТМИЗАЦИИ. ПРОГРАММИРОВАНИЕ В СРЕДЕ VISUAL C++

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

Библиотека БГУИР

Минск БГУИР 2009

УДК 004.432.2(076)
ББК 32.973.26-018.1я7
Н15

Р е ц е н з е н т :
доцент кафедры информационных технологий автоматизированных систем
учреждения образования «Белорусский государственный университет
информатики и радиоэлектроники»,
кандидат технических наук О. В. Герман

Библиотека БГУИР

Навроцкий, А. А.

Н15 Основы алгоритмизации. Программирование в среде Visual C++ : лаб. практикум по курсу «Основы алгоритмизации и программирования» для студ. 1 – 2-го курсов заоч. формы обуч. всех спец. БГУИР / А. А. Навроцкий, И. Н. Коренская, В. Л. Бусько. – Минск : БГУИР, 2009. – 48 с. : ил.

ISBN 978-985-488-474-5

Приведено 11 лабораторных работ на языке C++ в среде Microsoft Visual Studio с примерами выполнения; представлены индивидуальные задания; дана справочная информация.

УДК 004.432.2(076)
ББК 32.973.26-018.1я7

ISBN 978-985-488-474-5

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

СОДЕРЖАНИЕ

Библиотека БГУИР

Библиотека БГУИР

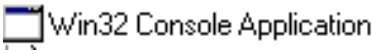
Лабораторная работа №1
СРЕДА ПРОГРАММИРОВАНИЯ VISUAL C++. ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

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:

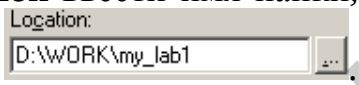
• выбрать в главном меню **File – New**;

• в открывшемся окне (закладка **Projects)** выбрать тип создаваемого проекта 

;

• в поле **Project Name** ввести имя проекта 

;

• в поле **Location** ввести имя папки, в которой будет размещен проект, и полный путь к ней 

. Папку также можно выбрать с помощью диалогового окна **Choose Directory**, для чего надо щелкнуть мышью по кнопке  ;

• щелкнуть мышью по кнопке **ОК;**

• в открывшемся окне мастера приложений **Win32 Console Application – Step 1 of 1** выбрать 

(пустой проект) и щелкнуть по кнопке 

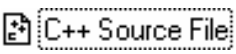
;

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

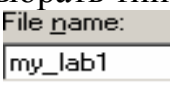
Для работы с консольным приложением необходимо создать **новый** или добавить **существующий** файл с кодом программы.

Для **создания нового файла** необходимо:

• выбрать **File – New**;

• в открывшемся окне (закладка **Files)** выбрать тип файла 

;

• в поле **File name** ввести имя файла: 

(желательно, чтобы вводимое имя совпадало с именем проекта);

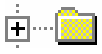
• щелкнуть мышью по кнопке **ОК.**

Для **добавления в проект существующего файла** с кодом программы необходимо:

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

Библиотека БГУИР

• в окне **Workspace** (закладка 

) щелкнуть правой кнопкой мыши по папке 

;

• в раскрывшемся меню выбрать пункт 

, в диалоговом окне 

указать добавляемый файл и щелкнуть мышью по кнопке **ОК.**

В папке проекта, как правило, размещены несколько файлов и одна вложенная папка. Файлы имеют следующее назначение:

- файл с расширением **dsw** (например *my_lab1.dsw*) – файл проекта, который объединяет все входящие в проект файлы;
- файл с расширением **cpp** (например *my_lab1.cpp*) – файл кода 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 math.lib**	**Описание**
$\|x\|$	**abs(x)**	Вычисление абсолютного значения (**только для целых чисел!**)
$\|x\|$	**fabs(x)**	Вычисление абсолютного значения *x*
$\sqrt{x}$	**sqrt(x)**	Вычисление квадратного корня из *x*
x^y	**pow(x, y)**	Возведение *x* в степень *y*
$\sin(x)$	**sin(x)**	Вычисление синуса *x*
$\mathrm{sh}(x) = (e^x - e^{-x})/2$	**sinh(x)**	Вычисление синуса гиперболического *x*
$\cos(x)$	**cos(x)**	Вычисление косинуса *x*
$\mathrm{ch}(x) = (e^x + e^{-x})/2$	**cosh(x)**	Вычисление косинуса гиперболического *x*
$\mathrm{tg}(x)$	**tan(x)**	Вычисление тангенса *x*
$\mathrm{tgh}(x)$	**tanh(x)**	Вычисление тангенса гиперболического *x*
$\arccos(x)$	**acos(x)**	Вычисление значения арккосинуса *x*
$\mathrm{arctg}(x)$	**atan(x)**	Вычисление значения арктангенса *x*
$\mathrm{arctg}(x/y)$	**atan2(x,y)**	Вычисление значения арктангенса двух аргументов *x* и *y*
e^x	**exp(x)**	Вычисление экспоненты числа *x*
$\ln(x)$	**log(x)**	Вычисление натурального логарифма *x*
$\lg_{10}(x)$	**log10(x)**	Вычисление десятичного логарифма *x*

Библиотека БГУИР

Математическая функция	Функция библиотеки math.lib	Описание
Округление к большему	**ceil(x)**	Функция возвращает действительное значение, соответствующее наименьшему целому числу, которое больше или равно *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*

Например:

```
double x, y,
double z = exp(1)+exp(x*x)+exp(2*pow(x,3));      // z = e +e^(x^2) +e^(2x^3)
double z1 = pow (x, pow (y, 4+pow (x, 1/4.)));   // z1 = x^(y^(4+4√x))
double z2 = fmod( 5, 2);        // или z2 = 5%2;   =>  z2 = {5/2} = 1
double z3 = ceil (5.6);         // => z3 = 6
double z4 = ceil (-5.1);        // => z4 = -5
double z5 = floor (5.6);        // => z5 = 5
double z6 = floor (-5.6);       // => z6 = -6
```

1.3. Пример выполнения работы

Написать программу вычисления линейного арифметического выражения

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

при $x = 2{,}45$; $y = -0{,}423 \cdot 10^{-2}$; $z = 1{,}232 \cdot 10^{3}$. **Ответ:** $h = 6{,}9465$.

Код программы

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

int main ()
{                                          // Начало главной функции
  double x, y, z, a, b, c, h;              // Объявление переменных

  cout << "Vvedite x:  ";                  // Ввод значений x, y и z
  cin >> x;
 cout << "Vvedite y:  ";
 cin >> y;
 cout << "Vvedite z:  ";
  cin >> z;
```

Библиотека БГУИР

```
  a = pow(x, 2*y) + exp(y-1);          // Вычисление выражения
  b = 1+x * fabs(y - tan(z));
  c = 10 * pow(x, 1/3.) - log(z);
  h = a / b + c;
  cout << "Result h= " << h << endl;    // Вывод результата
  return 0;                            // Завершение выполнения программы
}                                      // Конец главной функции
```

Замечания:

1. Для ввода значений переменных ***x***, ***y*** и **z** необходимо набрать с клавиатуры:

2.45 **(*x* = 2,45)**;

–0.423e-2 **(*y* = –0,423·10^{-2})**;

1.232e3 **(z = 1,232·10^{3})**.

2. В языке С++ при вычислении арифметических выражений происходит *автоматическое приведение типов*, следовательно, при делении целого значения на целое, результат будет *целым числом*. Например, при вычислениии «**1/3**» результат будет равен нулю, так как целая часть вычисленного выражения равна нулю. Для получения результата, имеющего дробную часть, необходимо, чтобы один из операндов имел действительный тип. Для вычисления можно использовать функцию *явного приведения типа*, а для констант достаточно *поставить точку после числа*, например, или «**1/3.**», или «**1./3.**», или «**1./3**».

Например:

```
int s, n;
double sr = static_cast<double> (s) / n;    // Явное приведение типа
double y = pow (x, 3/4.);                   // y = ∜(x^3) = x^(3/4)
```

3. Язык С чувствителен к регистру букв, т. е. прописные и строчные буквы воспринимаются как *разные* символы.

Например: count, Count, COUNT – разные идентификаторы.

4. При выводе информации для перехода на новую строку применяется манипулятор (функция управления выводом) **endl** или '**\n**', для выравнивания выводимой информации – '**\t**' (вставляет символ табуляции).

5. Главная функция **int main** () автоматически вызывается при запуске программы и возвращает операционной системе по окончании значение **0** (**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.

Библиотека БГУИР

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 на экран выводится соответствующее сообщение. В этом случае необходимо последовательно исправить все ошибки и компилировать проект снова. Если синтаксических ошибок нет, но результат выполнения программы неверный, то необходимо искать логические ошибки. Для этого следует использовать встроенный в систему отладчик (см. лабораторную работу №2).

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

1.4. Индивидуальные задания

Написать программу для вычисления значения выражения при заданных исходных данных. Сравнить полученное значение с указанным правильным результатом.

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

при $\boldsymbol{x}$ = 14,26; $\boldsymbol{y}$ = –1,22; $\mathbf{z}=3{,}5\cdot10^{-2}$. **Ответ:** $\boldsymbol{s}$ = 0,749155.

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

при $\boldsymbol{x}$ = –4,5; $\boldsymbol{y}=0{,}75\cdot10^{-4}$; $\mathbf{z}=-0{,}845\cdot10^{2}$. **Ответ:** $\boldsymbol{s}$ = –3,23765.

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

при $\boldsymbol{x}=3{,}74\cdot10^{-2}$; $\boldsymbol{y}$ = –0,825; $\mathbf{z}=0{,}16\cdot10^{2}$. **Ответ:** $\boldsymbol{s}$ = 1,05534.

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

при $\boldsymbol{x}=0{,}4\cdot10^{4}$; $\boldsymbol{y}$ = –0,875; $\mathbf{z}=-0{,}475\cdot10^{-3}$. **Ответ:** $\boldsymbol{s}$ = 1,98727.

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

при $\boldsymbol{x}$ = –15,246; $\boldsymbol{y}=4{,}642\cdot10^{-2}$; $\mathbf{z}$ = 21. **Ответ:** $\boldsymbol{s}$ = –182,038.

Библиотека БГУИР

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

при $x = 16{,}55\cdot10^{-3}$; $y = -2{,}75$; $z = 0{,}15$. **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 $x = 0{,}1722$; $y = 6{,}33$; $z = 3{,}25\cdot10^{-4}$. **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 $x = -2{,}235\cdot10^{-2}$; $y = 2{,}23$; $z = 15{,}221$. **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 $x = 1{,}825\cdot10^{2}$; $y = 18{,}225$; $z = -3{,}298\cdot10^{-2}$. **Ответ:** $s = 1{,}21308$.

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

при $x = 3{,}981\cdot10^{-2}$; $y = -1{,}625\cdot10^{3}$; $z = 0{,}512$. **Ответ:** $s = 1{,}26185$.

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

при $x = 6{,}251$; $y = 0{,}827$; $z = 25{,}001$. **Ответ:** $s = 0{,}712122$.

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

при $x = 3{,}251$; $y = 0{,}325$; $z = 0{,}466\cdot10^{-4}$. **Ответ:** $s = 4{,}23655$.

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

при $x = 17{,}421$; $y = 10{,}365\cdot10^{-3}$; $z = 0{,}828\cdot10^{5}$. **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 $x = 12{,}3\cdot10^{-1}$; $y = 15{,}4$; $z = 0{,}252\cdot10^{3}$. **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 $x = 2{,}444$; $y = 0{,}869\cdot10^{-2}$; $z = -0{,}13\cdot10^{3}$. **Ответ:** $s = -0{,}498707$.

Библиотека БГУИР

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 С 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. Оператор условной передачи управления if

Форматы оператора **if**:

1. Полная форма: **if (***логическое_выражение***)** *оператор_1*;
else *оператор_2*;

Если *логическое_выражение* истинно, то выполняется *оператор_1*, иначе – *оператор_2*.

2. Сокращенная форма: **if (***логическое_выражение***)** *оператор_1*;

Если *логическое_выражение* истинно, то выполняется *оператор_1*.

3. Вложенная форма:

if (*логическое_выражение_1***)** *оператор_1*;
else if (*логическое_выражение_2***)** *оператор_2*;
else *оператор_3*;

Если *логическое_выражение_1* истинно, то выполняется *оператор_1*, иначе, если *логическое_выражение_2* истинно, то выполняется *оператор_2*, иначе выполняется *оператор_3*.

2.3. Оператор множественного выбора switch

Общая форма оператора:

```
switch (переменная_выбора)
 {      case const_1: операторы_1;   break;
                  …
        case const_N: операторы_N;   break;
        default:      операторы_N+1;
 }
```

переменная_выбора, **const_1, …, const_N** – константа, переменная или выражение *целого*, *символьного* или *логического* типа.

Библиотека БГУИР

При использовании оператора **switch** сначала анализируется *переменная_выбора* и проверяется, совпадает ли ее значение со значением одной из констант **const_1, …, const_N**. При совпадении выполняются операторы этого **case**. Конструкция **default** (может отсутствовать) выполняется, если результат выражения не совпал ни с одной из констант.

2.4. Пример выполнения работы

Написать программу вычисления выражения $s=\begin{cases}|\cos(x)|+\ln(y), & |xy|>10,\\ e^{2x+y}, & 3<|xy|\le 10,\\ \sqrt{|x|}+2\mathrm{tg}(y), & \text{иначе.}\end{cases}$

Предусмотреть вывод информации о выбранной ветви вычислений.

Код программы

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

int main()
{
  double x, y, s;

  cout << "Vvedite x:  ";
  cin >> x;
  cout << "Vvedite y:  ";
  cin >> y;

  double f_xy = fabs(x*y);
  if (f_xy > 10) {                                        //|xy|>10
                  s = fabs(cos(x)) + log(y);
                  cout << "1 vetv. Result = " << s << endl;
                }
    else if (f_xy>3 && f_xy<=10)                          //3<|xy|≤10
                          { s = exp(2*x+y);
                            cout << "2 vetv. Result = " << s << endl;
                           }
               else {                                     // Иначе
                      s = sqrt(fabs(x)) + 2*tan(y);
                      cout << "3 vetv. Result = " << s << endl;
                     }
  return 0;
}
```

2.5. Индивидуальные задания

Составить программу вычисления выражения согласно указанному варианту. Предусмотреть вывод информации о выбранной ветви вычислений.

Библиотека БГУИР

1. $s=\begin{cases}(x+y)^2-\sqrt[3]{|x|}, & xy>0,\\ (x+y)^2+\sin(x), & xy<0,\\ (x+y)^2+y^3, & \text{иначе.}\end{cases}$

2. $s=\begin{cases}\ln(x)+\sqrt[3]{|y|}, & x/y>0,\\ \ln|x/y|\cdot(x+y)^3, & x/y<0,\\ \left(x^2+y\right)^3, & \text{иначе.}\end{cases}$

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

4. $s=\begin{cases}\sqrt[3]{|x-y|}+\operatorname{tg}(x), & x>y,\\ (y-x)^3+\cos(x), & x<y,\\ (y+x)^2+x^3, & \text{иначе.}\end{cases}$

5. $s=\begin{cases}y\sqrt{|x|}+3\sin(x), & x>y,\\ x\sqrt{|x|}, & x<y,\\ \sqrt[3]{|x|}+x^3/y, & \text{иначе.}\end{cases}$

6. $s=\begin{cases}e^{x-|y|}, & 0{,}5<xy<10,\\ \sqrt[3]{|x+y|}, & 0{,}1<xy<0{,}5,\\ 2x^2, & \text{иначе.}\end{cases}$

7. $s=\begin{cases}e^{-x}, & 1<xb<10,\\ \sqrt[3]{|x+4y|}, & 12<xb<40,\\ y\cdot x^2, & \text{иначе.}\end{cases}$

8. $s=\begin{cases}\left(x^2+y\right)^3, & x/y<0,\\ \ln|x/y|+x/y, & x/y>0,\\ \sqrt[3]{|\sin(y)|}, & \text{иначе.}\end{cases}$

9. $s=\begin{cases}2x^3+3y^2, & x>|y|,\\ |x-y|, & 3<x<|y|,\\ \sqrt[3]{|x-y|}, & \text{иначе.}\end{cases}$

10. $s=\begin{cases}\ln(|x|+|y|), & |xy|>10,\\ e^{x+y}, & |xy|<10,\\ \sqrt[3]{|x|}+y, & \text{иначе.}\end{cases}$

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

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

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

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

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

Библиотека БГУИР

Лабораторная работа №3
ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

3.1. Оператор цикла с параметром 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 **true** (**1**), 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 с предусловием while

Общий вид оператора:

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

Операторы *тела цикла* повторяются до тех пор, пока *условие* истинно.

3.3. Оператор цикла с постусловием do

Общий вид оператора:

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

Операторы *тела цикла* повторяются до тех пор, пока *условие* истинно.

3.4. Операторы перехода

Оператор **break** прекращает выполнение ближайшего к нему цикла или оператора **switch**.

Оператор **continue** передает управление на проверку условия циклов **while** и **do while**, либо на *инкрементирующее_выражение* цикла **for**.

Библиотека БГУИР

3.5. 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 код 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 следует поместить курсор в нужную строку и нажать **F9**. Точка прерывания обозначается красным кружком на специальном поле, расположенном слева от окна кода 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 применяется клавиша **F5** (выполнение программы до следующей точки прерывания) или используются клавиши для пошаговой отладки.

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

Для контроля за значениями переменных удобно использовать следующий способ: подвести указатель мыши к нужной 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.

3.6. Пример выполнения работы

Написать программу вывода на экран таблицы значений функции $\sum_{k=0}^{20} \frac{2x^k}{\cos^n(x)}$ *для x, изменяющегося от a* = 0,1 *до b* = 1 *с шагом h* = 0,1.

Код программы

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>

int main()
{
  double  s, x, a, b, h;
  int     k, i;

  cout << "Vvedite a, b, h, k:" << endl;
  cin >> a >> b >> h >> k;                    // Ввод значений: 0.1  1  0.1  20
```

Библиотека БГУИР

```
    x = a;
    do                                          // Начало цикла по  x
    {
      s = 2;                                    // Начальное значение при k=0
      for (i = 1; i <= k; i++)                  // Вычисление суммы  Σ_{k=1}^{20} 2x^k / cos^n(x)
          s += 2*pow(x, i) / pow(cos(x), i);
      cout << setw(15) << x << setw(15) << s << endl;      // Вывод таблицы
      x += h;                                   // Изменение значения x на величину шага h
    }
      while (x <= b + h/2);                     // Проверка условия продолжения цикла по x
  cout << endl;                                 // Переход на новую строку
  return 0;
}
```

3.7. Индивидуальные задания

Вывести на экран таблицу значений функции $y(x)$ *для* x*, изменяющегося от* $a = 0{,}1$ *до* $b = 1{,}2$ *с шагом* $h = 0{,}1$.

1. $y = \sum_{n=1}^{20} \frac{x^{n-1}}{2n+1}$.

2. $y = \sum_{n=0}^{20} \frac{(2x)^n}{n+1}$.

3. $y = \sum_{n=1}^{20} \frac{x^{n-1}}{\sin(nx)}$.

4. $y = \sum_{n=1}^{20} \frac{n^2+1}{n}\left(\frac{x}{2}\right)^n$.

5. $y = \sum_{n=1}^{20} \frac{\cos\left(n \cdot \frac{\pi}{4}\right)}{n+1} x^n$.

6. $y = \sum_{n=1}^{20} \frac{x^{2n-2}}{2n+1}$.

7. $y = \sum_{n=0}^{20} \frac{x^{2n}}{\cos(nx)}$.

8. $y = \sum_{n=1}^{20} \frac{2n^2+1}{2n} x^{2n-2}$.

9. $y = \sum_{n=1}^{20} \frac{2n+1}{\sin(nx)} x^{n-1}$.

10. $y = \sum_{n=0}^{20} \frac{\cos^n(x)}{2n+1}$.

11. $y = \sum_{n=1}^{20} \frac{n \cdot x^{n-1}}{\sin(2n+x)}$.

12. $y = \sum_{n=1}^{20} \frac{(1+x)^{n-1}}{\left|\sin^n(x)\right|}$.

13. $y = \sum_{n=1}^{20} \frac{x^{2n-2}}{4\cos(nx^2)}$.

14. $y = \sum_{n=1}^{20} \frac{n^2}{(2n+1)} x^{n-1}$.

15. $y = \sum_{n=0}^{20} \frac{\cos\left(n\frac{\pi}{4}\right)}{n^2} x^n$.

Библиотека БГУИР

Лабораторная работа №4
ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ОДНОМЕРНЫХ МАССИВОВ

4.1. Одномерные статические массивы

Массив – это набор данных одинакового типа, расположенных в непрерывной области памяти таким образом, чтобы по индексу элемента можно было вычислить адрес его значения:

адрес(a[i]) = адрес(a[0]) + i*k,

где ***i*** – индекс элемента массива;

k – количество байт, отводимых под элемент массива.

Для доступа к элементу массива необходимо указать его имя и индекс (порядковый номер элемента в масиве):

имя_массива [индекс]

В программе одномерный массив объявляется следующим образом:

***тип имя_массива* [*размер*];**

Пример *декларации 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.*

Пример 1. *Упорядочить элементы массива по возрастанию их значений, т. е. для всех элементов массива должно выполняться условие:* $a_i < a_{i+1}$.

```
  …
for (k=1; k<n; k++)          // k – номер просмотра массива
   for (i=0; i<n-k; i++)     // Просмотр элементов массива
    if (a[i] > a[i+1])       // Сравнение элементов массива
          {  temp=a[i];      // Перестановка элементов ai и ai+1,
             a[i]=a[i+1];    // если они стоят неправильно
             a[i+1]=temp;
          }
```

Пример 2*. Удалить из одномерного массива все отрицательные элементы.*

Для решения данной задачи необходимо выполнить следующие действия:

```
  …
for (i=0; i<n; i++)
  if (a[i]<0)              // Если найден отрицательный элемент, то
    {
      for (j=i+1; j<n; j++)   // сдвинуть все элементы, стоящие
                 a[j-1]=a[j]; // после удаляемого на одну позицию
      n--;                    // Уменьшение размера массива
       i--;                   // Возврат к предыдущему индексу
     }
```

Библиотека БГУИР

Пример 3. *Даны одномерные упорядоченные по возрастанию массивы:* ***X*** *размером* ***n*** *элементов и* ***Y*** *размером* ***m*** *элементов. Объединить элементы этих массивов в массив* ***Z*** *так, чтобы и он оказался упорядоченным по возрастанию.*

```
…
k=i=j=0;
while(i<n && j<m) {
                    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<m)  {
               c[k]=b[j];
               j++;
               k++;
            }
```

4.2. Пример выполнения работы

Составить программу поиска минимального и максимального элементов одномерного массива и их индексов.

```
#include <iostream.h>
#include <math.h>
int main()
{
  int a[10], i, n, min, imin, max, imax;
  cout << "Vvedite razmer: ";                    // Ввод одномерного массива
  cin >> n;
  for (i=0; i<n; i++)
     {
       cout << "Vvedite a[" << i << "]= " ;
       cin >> a[i];
     }
  cout << "Massiv a:" << endl;                   // Вывод одномерного массива
  for (i=0; i<n; i++)
     cout << a[i] << "  " ;
```

Библиотека БГУИР

```
  cout << endl;
  min=max=a[0];
  imin=imax=0;
  for (i=1; i<n; i++)
        if (a[i]<min)  { min=a[i];
                         imin = i;
                       }
              else
                   if (a[i]>max) { max=a[i];
                                   imax = i;
                                 }
  cout << "Max = " << max << "   i=" << imax << endl;
  cout << "Min = " << min << "    i=" << imin << endl;
  return 0;
}
```

4.3. Индивидуальные задания

Ввести одномерный статический массив из k чисел. Выполнить в соответствии с номером варианта индивидуальное задание и вывести на экран исходные данные и полученный результат.

1. Преобразовать массив следующим образом: все отрицательные элементы перенести в начало массива, сохранив исходное взаимное расположение как среди отрицательных, так и среди остальных элементов.

2. Расположить элементы массива в обратном порядке.

3. Найти и поменять местами элементы, имеющие минимальное и максимальное значения в массиве.

4. Определить, упорядочены ли элементы массива по убыванию.

5. Вывести все неповторяющиеся элементы массива.

6. Сдвинуть элементы массива циклически на *n* позиций *влево*.

7. Сдвинуть элементы массива циклически на *n* позиций *вправо*.

8. Удалить минимальный и максимальный элементы массива.

9. Сформировать два новых массива: в первый записать отрицательные элементы исходного массива, во второй – все остальные.

10. Определить, симметричен ли массив, т. е. читается ли он одинаково слева направо и справа налево.

11. Найти количество элементов массива, отличающихся от среднего значения элементов массива не более чем на *3*.

12. Определить количество инверсий в массиве (таких пар элементов, в которых большее значение находится слева от меньшего).

13. Определить количество элементов, значение которых больше среднего значения всех элементов массива.

14. Удалить элементы, значение которых меньше среднего значения всех элементов массива.

15. Удалить из массива повторяющиеся элементы.

Библиотека БГУИР

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 С существует *три вида указателей*:

1. Указатель на объект *известного типа* содержит адрес объекта определенного типа.

Например: **int** *ptr;

2. Указатель типа **void** применяется, если тип объекта заранее не определен.

Например: **void** *vptr;

3. Указатель на *функцию* – это адрес, по которому передается управление при вызове функции.

Например: void (***func**)(int);

***func** указатель на функцию, принимающую аргумент **int** и не возвращающую никаких значений.

5.2. Операции над указателями

К указателям можно применять две *унарные операции*:

1. & (взятие адреса).

Указатель получает адрес переменной. Данная операция применима к переменным, под которые выделен соответствующий участок памяти.

```
Например:  int *ptr, var=1;        // ptr – указатель, var – переменная
           ptr = &var;             // В ptr заносится адрес var
```

2. * (операция разадресации).

Предназначена для доступа к значению, расположенному по данному адресу.

```
*ptr = 9;      // В ячейку памяти с адресом ptr записывается значение 9
var = *ptr;    // Переменной var присваивается значение,
               // расположенное по адресу ptr
```


Библиотека БГУИР

Над указателями можно выполнять арифметические операции *сложения*, *инкремента* (увеличения на 1), *вычитания*, *декремента* (уменьшения на 1) и операции *сравнения* (**>**, **>=**, **<**, **<=**, **==**, **!=**). При выполнении арифметических операций с указателями автоматически учитывается размер данных, на которые он указывает.

Например:

```
ptr++;      // Сдвиг указателя ptr на один элемент вперед
(*ptr)++;  // (или ++*ptr;) Увеличение на 1 значения переменной,
           // на которую указывает указатель ptr
*ptr = NULL;  // Очистка указателя ptr
```

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

В языке С++ 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 заданного 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.

Пример *работы с одномерным динамическим массивом*:

```
int *a;              // Объявление указателя a
a = new int[n];   // Выделение n блоков памяти целого типа
      …           // Работа с массивом a
delete [] a;      // 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 выделятся память *под указатели*, а затем под соответствующие этим указателям *строки*. Освобождение выделенной памяти происходит *в обратном порядке*.

5.4. Пример выполнения работы

Написать программу перестановки минимального и максимального элементов двумерного массива размером N×M. Память для массива выделить динамически.

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>

int main()
```

Библиотека БГУИР

```
{
  double  **a, tmp;
  int i, j, n, m, imin, jmin, imax, jmax;

  cout << "Vvedite razmer: n, m" << endl;
  cin >> n >> m;

  a = new double*[n];              // Выделение памяти под массив указателей
  for(i=0; i<n; i++)               // Выделение памяти под соответствующие
        a[i] = new double[m];      // этим указателям строки матрицы

  for (i=0; i<n; i++)              // Ввод двумерного массива
    for (j=0; j<m; j++)
     {
        cout << "Vvedite a[" << i << "][" << j << "]: " ;
        cin >> a[i][j];
     }

  cout << "Massiv A:" << endl;     // Вывод двумерного массива
  for (i=0; i<n; i++)
     {
        for (j=0; j<m; j++)
           cout << setw (9) << a[i][j] << "  ";
        cout << endl;
     }

  imin=jmin=imax=jmax=0;           // Поиск индексов минимального
  for (i=0; i<n; i++)              // и максимального элементов массива
     for (j=0; j<m; j++)
        {
           if (a[i][j]<a[imin][jmin]) { imin=i;
                                        jmin=j;
                                      }
           if (a[i][j]>a[imax][jmax]) { imax=i;
                                        jmax=j;
                                      }
        }
  tmp = a[imin][jmin];             // Перестановка элементов
  a[imin][jmin] = a[imax][jmax];
  a[imax][jmax] = tmp;

  cout << "Result :" << endl;      // Вывод результата
  for (i=0; i<n; i++)
     {
        for (j=0; j<m; j++)
```

Библиотека БГУИР

```
            cout << setw (9) << a[i][j] << "  ";
        cout << endl;
    }
    for(i=0; i<n; i++)              // Освобождение выделенной памяти
            delete [] a[i];
    delete []a;
    a = NULL;

    return 0;
}
```

5.5. Индивидуальные задания

Ввести матрицу размером N×M. Память для массива выделить динамически. Выполнить в соответствии с номером варианта индивидуальное задание и вывести на экран исходные данные и полученный результат.

1. Определить количество положительных элементов, расположенных ниже побочной диагонали матрицы.

2. Определить количество отрицательных элементов, расположенных выше главной диагонали матрицы.

3. Определить сумму отрицательных элементов, расположенных выше побочной диагонали матрицы.

4. Определить произведение положительных элементов, расположенных ниже главной диагонали матрицы.

5. Определить сумму элементов, расположенных на главной диагонали матрицы, и произведение элементов, расположенных на побочной диагонали матрицы.

6. Определить количество четных элементов, расположенных на главной и побочной диагоналях.

7. Найти максимальный среди элементов, лежащих ниже побочной диагонали.

8. Найти минимальный среди элементов, лежащих выше главной диагонали.

9. Найти максимальный среди элементов, лежащих выше побочной диагонали.

10. Найти минимальный среди элементов, лежащих ниже главной диагонали.

11. Найти в каждой строке матрицы максимальный элемент.

12. Найти в каждом столбце матрицы минимальный элемент.

13. Найти сумму элементов, расположенных в четных (по номеру) строках матрицы.

14. Найти произведение элементов, расположенных в нечетных (по номеру) столбцах матрицы.

15. Подсчитать сумму четных элеметнов и произведение нечетных элементов матрицы.

Библиотека БГУИР

Лабораторная работа №6
ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СТРОК

6.1. Объявление строк

Объявление строки аналогично объявлению массива:

char *имя_строки* [*размер*];

Важной особенностью такого объявления является то, что строка должна *обязательно заканчиваться нулевым символом* **'\0'** – (*нуль-терминатор*). Длина строки (*размер*) 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 **0**, если **st1 > st2** – результат равен **1**.

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

char *strchr (st, ch) – 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 **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**).

Библиотека БГУИР

6.3. Пример выполнения работы

Пример 1. *Задана строка символов, в которой слова отделяются друг от друга одним или несколькими пробелами. Вывести на экран все слова этой строки.*

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

int main()
{
   char st[100], sl[100];
   int k = 0, i;

   puts ("Vvedite stroku");
   gets (st);

   strcat (st," ");
   puts ("Slova: ");
   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;
         }

   return 0;
}
```

Пример 2. *Подсчитать количество различных символов в строке.*

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

int main()
{
 char st[100], tmp[100];
 int n = 0;
```

Библиотека БГУИР

```
char *ch = NULL;
tmp[0] = '\0';
puts("Vvedite stroku: ");
gets(st);
for(int i=0; st[i] != '\0'; i++)
  {
     ch = strchr (tmp, st[i]);
     if (ch == NULL)
      {
        n++;
        tmp[n-1] = st[i];
        tmp[n] = '\0';
      }
  }
 cout << "Chislo razl. simvolov = " << n << endl;
 return 0;
}
```

6.4. Индивидуальные задания

Вводится строка, каждое слово которой отделяется от других слов одним или несколькими пробелами. Выполнить в соответствии с номером варианта индивидуальное задание и вывести на экран полученный результат.

1. Найти количество слов, состоящих из *пяти* символов.

2. Найти самое *короткое* слово в строке.

3. Определить количество символов во *втором* слове.

4. Найти слова с *четным* количеством символов.

5. Найти порядковый номер слова *максимальной* длины.

6. Найти слова, содержащие подстроку «*sas*».

7. Найти все *числа* в строке. Каждое число вывести в отдельной строке.

8. Подсчитать количество слов, начинающихся с буквы «*a*».

9. Подсчитать количество слов, заканчивающихся на букву «*z*».

10. Вывести на экран порядковый номер слова *максимальной* длины и номер позиции в строке, с которой оно начинается.

11. Найти самое *длинное* слово в строке.

12. Определить порядковый номер слова, накрывающего *k*-ю позицию (если на *k*-ю позицию попадает пробел, то номер предыдущего слова).

13. 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. Найти слова, содержащие букву «*s*».

15. Найти порядковый номер слова *минимальной* длины.

Библиотека БГУИР

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 указывается ее имя и 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

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

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

int main ()
{
  struct strc              // Объявление структуры strc
  { char fio[40];
    char ngr[7];
    int otc[4];
    double sb;
  } mstud[100];            // Объявление массива структур mstud
  int nst, i, j;
  cout << "Vvedite kol-vo studentov" << endl;
  cin >> nst;
  for (i=0; i < nst; i++)        // Ввод информации о студентах
    {
      cout << "Vvedite FIO: ";
```

Библиотека БГУИР

```
            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++)                    // Ввод четырех оценок
              {                                    // за последнюю сессию
                cin >> mstud[i].otc[j];
                mstud[i].sb += mstud[i].otc[j] / 4.;   // Вычисление
              }                                    // среднего балла студента
            cout << endl;
        }
      strc stemp;
      for (i=0; i < nst-1; i++)              // Сортировка по среднему баллу
        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];        // Перестановка структур
                  mstud[i] = mstud[j];
                  mstud[j] =stemp;
                }
      for (i=0; i < nst; i++)
        if (!strcmp (mstud[i].ngr, "610205"))        // Вывод информации
            cout << mstud[i].fio << "  " << mstud[i].ngr << "  "
                 << mstud[i].sb << endl;
      return 0;
    }
```

7.3. Индивидуальные задания

Создать массив структур, содержащий информацию согласно варианту индивидуального задания. Выполнить задание и вывести на экран полученный результат.

1. В магазине сформирован список постоянных клиентов, который включает ФИО, домашний адрес покупателя и размер предоставляемой скидки. Вывести список покупателей, имеющих *5* %-ную скидку.

2. Список товаров, имеющихся на складе, включает в себя наименование товара, количество единиц товара, цену единицы и дату поступления товара на склад. Вывести список товаров, стоимость которых превышает *100 000* рублей.

3. Для получения места в общежитии формируется список студентов, который включает ФИО студента, номер группы, средний балл, доход на члена

Библиотека БГУИР

семьи. Вывести фамилии студентов, у которых доход на члена семьи *меньше двух* минимальных заработных плат.

4. В справочной автовокзала имеется 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 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. Вывести размер заработной платы каждого сотрудника.

7. Информация об участниках спортивных соревнований содержит название страны, название команды, ФИО игрока, игровой номер, возраст, рост и вес. Вывести фалимии спортсменов, возраст которых *больше 20* лет.

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

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

10. Информация о сотрудниках содержит ФИО, номер отдела, должность, стаж работы на предприятии. Вывести список сотрудников заданного отдела, имеющих стаж работы на предприятии *более 20* лет.

11. Ведомость абитуриентов содержит ФИО, адрес, оценки по *трем* предметам. Определить средний балл абитуриентов, проживающих в городе *Минске*.

12. В справочной аэропорта имеется расписание вылета самолетов. Для каждого рейса указаны его номер, тип самолета, пункт назначения, время вылета. Вывести все номера рейсов, вылетающих в заданный пункт назначения.

13. У администратора железнодорожных касс имеется информация о свободных местах в поездах на текущие сутки в следующем виде: пункт назначения, время отправления, число свободных мест. Вывести информацию о числе свободных мест в поездах, следующих до заданного пункта назначения.

14. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит ФИО абитуриента и его оценки. Определить средний балл по университету и вывести список абитуриентов, средний балл которых выше среднего балла по университету.

15. В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждая квитанция содержит наименование изделия, дату приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Вывести информацию об изделиях, ремонт которых еще не выполнен.

Библиотека БГУИР

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**).

Список_параметров представляет собой список конструкций следующей формы:

тип параметра имя параметра

Например:

```
int Sum (int a, double b, char c)
void Prints (char c, int f)
```

Если функция не получает никаких данных, то скобки остаются пустыми:

```
int Mem ();
```

Как правило, помимо описания функции, в программу вставляется *прототип* функции (ее предварительное объявление). Прототип аналогичен заголовку функции, только на конце его ставится точка с запятой, а имена формальных параметров не указываются (остаются только типы):

```
int Sum (int, double, char);
```

Правила оформления тела функции такие же, как и любого другого участка программы. Все объявления носят локальный характер, т. е. объявленные переменные доступны только внутри функции.

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 полученное значение, то выход осуществляется оператором

return *выражение*;

Библиотека БГУИР

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, по ссылке или по указателю.

8.2.1. Передача параметров по значению

В момент обращения к функции в памяти создаются временные переменные с именами, указанными в списке параметров, в которые копируются значения фактических (передаваемых в функцию) параметров. После завершения работы функции временные переменные удаляются из памяти.

Пример. *Вычислить сумму двух переменных* **x** *и* **y.**

```
double Sum(double a, double b)
 {
   return a+b;          // Вычисление и передача результата
 }
   …
   s = Sum (x, y);          // Вызов функции Sum
```

Функция Sum не может изменить значения используемых при вызове функции переменных **x** и **y**, так как работает только с их *локальными копиями*.

8.2.2. Передача параметров по ссылке

При передаче параметров по ссылке передается **адрес** соответствующей переменной, а не ее значение. Для получения адреса используется операция разадресации («**&**»).

Пример. *Поменять местами значения двух переменных* **x** *и* **y**.

```
void Swap_Ref (double &a, double &b)
 {
   double tmp = a;
   a = b;
   b = tmp;
 }
   …
Swap_Ref (x, y);          // Вызов функции Swap_Ref
```

При таком способе передачи параметры **a** и **b** будут инициализированы в качестве псевдонимов переменных-аргументов **x** и **y**. Поэтому любые изменения параметров **a** и **b** будут приводить к соответствующему изменению переменных **x** и **y**.

Библиотека БГУИР

8.2.3. Передача параметров по указателю

Так же как и при передаче параметров по ссылке, в данном способе используется не значение соответствующей переменной, а ее **адрес**. Отличие от предыдущего способа состоит в том, что используется **операция косвенной адресации** (*).

Пример. *Поменять местами значения двух переменных* **x** *и* **y**:

```
void Swap_Ptr (double *a, double *b)
{
  double tmp= *a;
  *a = *b;
  *b = tmp;
}
  …
Swap_Ptr (&x, &y);          // Вызов функции Swap_Ptr
```

Функция **Swap_Ptr** требует явного указания адресов при своем вызове (&x, &y) и явного их разыменования в функции (*a и *b).

8.3. Перегрузка функций

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*.

Пример. *Написать функцию, которая суммирует или* ***два****, или* ***три*** *целых числа, или* ***все*** *элементы целочисленного массива.*

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

// Прототипы функций
int Sum(int, int);
int Sum(int, int, int);
int Sum(int[], int);

int main()
{
  int mas[] = {3,2,4,9,5,8};        // Инициализация массива mas
  cout << Sum(5, 3)     << endl;
  cout << Sum(5, 3, 11) << endl;
  cout << Sum(mas, 6) << endl;
  return 0;
}

int Sum(int a, int b)          // Функция суммирования двух чисел
{
  return a+b;
```

Библиотека БГУИР

```
}
int Sum(int a, int b, int c)   // Функция суммирования трех чисел
{
  return a+b+c;
}
int Sum(int a[], int n)  // Функция суммирования элементов массива
{
   int s = 0;
   for (int i=0; i<n; i++)
       s+= a[i];
   return s;
}
```

8.4. Указатель на функцию

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, имеется функция

```
double y (double x, int n) {тело функции}
```

Объявляем указатель на функцию:

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

Присвоим указателю **fun** адрес функции **y**:

```
fun = y;
```

Функцию через указатель можно вызывать так:

```
x = fun (t, m);
```

или

```
x = (*fun) (t, m);
```

8.5. Пример выполнения работы

Составить программу вывода на экран таблицы значений функции $\sum_{k=0}^{20} \frac{2x^k}{\cos^n(x)}$ *для x, изменяющегося от a* = 0,1 *до b* = 1 *с шагом h* = 0,1. *Написать фукцию табуляции и использовать ее для вывода значений функции y*(*x*).

```
#include <iostream.h>
#include <math.h>
#include <iomanip.h>

typedef double (*uf)(double, int);

void Tabl (double, double, double, uf);
double Y (double, int);
```

Библиотека БГУИР

```
int main()
{
   cout << setw(8) <<"x"<< setw(15) <<"y(x)"<< setw(10) << endl;
   Tabl (0.1, 1, 0.1, Y);
   cout << endl;
   return 0;
}

void Tabl (double a, double b, double h, uf fun)
{
   int k=0;
   double sum;
   for (double x=a; x < b+h/2; x+=h)
      {
        sum = fun(x, 20);
        cout << setw(8) << x << setw(15) << sum << endl;
      }
}

double Y (double x, int k)
{
   double    s = 2;
   for (int i=1; i <= k; i++)
         s += 2*pow(x, i) / pow(cos(x), i);
   return s;
}
```

8.6. Индивидуальные задания

Вывести на экран таблицу значений функции $y(x)$ *для* x*, изменяющегося от* $a = 0{,}1$ *до* $b = 1{,}2$ *с шагом* $h = 0{,}1$ (*см. лабораторную работу* №3, *п.* 3.7). *Вычисление* $y(x)$ *оформить в виде функции.*

Библиотека БГУИР

Лабораторная работа №9
ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ФАЙЛОВ

9.1. Организация работы с файлами

Различают два вида файлов: *текстовые* и *двоичные* (*бинарные*).

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 указателем, который 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.

9.2. Функции для работы с файлами

Перед началом работы с файлом его необходимо открыть функцией

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

которая связывает файл с потоком и возвращает указатель на открытый файл. *Имя_файла* и *режим_открытия* – указатели на строки символов, содержащие соответственно путь к файлу, его имя и 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 читать и записывать в файл, то добавляется символ «**+**». При возникновении ошибки во время открытия файла функция **fopen** возвращает значение **NULL**.

После завершения работы с файлом его необходимо закрыть функцией

int fclose (FILE * *указатель_на _файл***);**

которая закрывает поток, открытый с помощью вызова **fopen ()**, и записывает в файл данные, оставшиеся в дисковом буфере. Результатом работы функции

Библиотека БГУИР

может быть значение нуля (успешная операция закрытия) или EOF (ошибка). Доступ к файлу после выполнения функции будет запрещен.

Функция

int fcloseall (void);

закрывает все открытые файлы и 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*);

Библиотека БГУИР

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_данное*,
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 значение без знака.

Функция

size_t fread (void * *считываемое_данное*,
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* (логический номер файла для заданного потока) указанного файла.

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** или **0** – начало файла, **SEEK_CUR** или **1** – текущая позиция, **SEEK_END** или **2** – конец файла.

9.3. Пример выполнения работы

Написать программу формирования файла, содержащего экзаменационную ведомость студентов: фамилию и оценки по математике и программированию. Предусмотреть возможность чтения из файла. Вывести список студентов, сдавших экзамен по программированию с оценкой 9, и записать эту информацию в текстовый файл.

Библиотека БГУИР

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

FILE *fl;
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();                 // Создание меню
void Nnf();                 // Ввод имени файла
void Newf();                // Создание нового файла
void Spisok();              // Формирование файла
void Opf();                 // Открытие файла
void Resc();                // Вывод результата на экран
void Resf();                // Вывод результата 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: puts("Viberite pravilno!");
      }
      puts ("Press any key to continue");
      getch ();             // Ожидание нажатия любой клавиши
      system ("cls");       // Очистка экрана
   }
}
```

Библиотека БГУИР

```
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;                          // Ввод выбранного пункта меню
   return i;
}

void Nnf()                            // Ввод имени файла
{
   cout << "Vvedite file name" << endl;
   cin >> name;
}

void Newf()                           // Создание нового файла
{
   if ((fl = fopen(name,"wb")) == NULL)
      {
         cout << "Oshibka pri sozdanii" << endl;
         exit(1);
      }
      cout << "OK" << endl;
      fclose(fl);
}

void Spisok()                         // Ввод данных в файла
{
   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: ";
```

Библиотека БГУИР

```
            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()                          // Открытие бинарного файла
{
    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()                         // Вывод результата на экран
{
    for (int i=0; i<nst; i++)
        if (stud[i].oaip == '9')
                cout << stud[i].fio << endl;
}

void Resf()                         // Вывод результата в текстовый файл
{
    char namet[30];
    FILE *ft;
```

Библиотека БГУИР

```
    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 == '9')
      {
        strcpy (s, stud[i].fio);
        strcat (s, "\n");              // Добавление разделителя строк
        fputs (s, ft);
      }
    fclose(ft);
}
```

9.4. Индивидуальные задания

Написать программу формирования файла, содержащего данные согласно варианту индивидуального задания. В программе предусмотреть сохранение вводимых данных в файл и возможность чтения из ранее сохраненного файла. Вывести результаты на экран и в текстовый 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 товара. Вывести список товаров, стоимость которых превышает *1 000 000* рублей.

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 заработных плат.

3. В справочной автовокзала имеется расписание движения автобусов. Для каждого рейса указаны 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 проработанных часов за месяц, почасовой тариф. Вывести размер заработной платы каждого сотрудника фирмы.

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

Библиотека БГУИР

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

7. На заводе производится несколько наименований деталей. Сведения о деталях, выпущенных за последний месяц, включают номер цеха, наименование детали, количество выпущенных деталей. Вывести информацию о продукции, выпущенной заданным цехом.

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

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

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

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 марку телевизора, дату приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Вывести информацию о невыполненных на текущий момент заказах.

14. На АТС информация о разговорах содержит номер телефона абонента, время разговора и тариф. Вывести для заданного абонента сумму оплаты 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.

Библиотека БГУИР

Лабораторная работа №10
СОРТИРОВКА ПО КЛЮЧУ ОДНОМЕРНЫХ МАССИВОВ СТРУКТУР

10.1. Сортировка массивов

Сортировка – упорядочивание элементов массива по возрастанию или по убыванию.

10.1.1. Метод пузырька

Последовательно сравниваются рядом стоящие элементы массива и, если условие упорядоченности не выполняется, то эти элементы переставляются. За один цикл в необходимую позицию перемещается один элемент массива и из дальнейшего рассмотрения исключается.

```
void S_Puz(int a[], int n)          // Сортировка по возрастанию
{   int i , j, t;
   for(i=1; i < n; i++)
     for( j=n-1; j >= i; j--)       // Перебор элементов
        if (a[j-1] > a[j])
                 {  t = a[j-1];     // Перестановка элементов
                    a[j-1] = a[j];
                    a[j] = t;
                 }
}
```

10.1.2. Сортировка выбором

В массиве выбирается минимальный элемент, помещается в первую позицию и из рассмотрения исключается. Затем аналогичным образом перемещаются элементы в оставшейся части массива. Процесс повторяется до тех пор, пока все элементы не будут перемещены в необходимые позиции.

```
void S_Vb(int a[], int n)
{  int imin, i, j, t;
   for(i = 0; i < n-1; i++)          // Перебор элементов
   {
      imin = i;
      for (j = i+1; j < n; j++)     // Поиск минимального элемента
         if (a[imin] > a[j])
                              imin = j;
            if (imin != i) {    // Перемещение минимального элемента
                              t = a[imin];
                              a[imin] = a[i];
                              a[i] = t;
                            }
   }
}
```

Библиотека БГУИР

10.1.3. Сортировка вставками

Сначала сортируются два первых элемента массива. Затем алгоритм вставляет третий элемент в необходимую позицию по отношению к первым двум элементам. После этого четвертый и так далее. Процесс повторяется до тех пор, пока не будут вставлены все элементы.

```
void S_Vst (int a[], int n)
{
   int i, j, t;
   for(i=1; i<n; i++)                          // Перебор элементов
    {
     t = a[i];                                 // Выбор  элемента
     for(j = i-1; j >= 0 && t < a[j]; j--)     // Поиск необходимой позиции
        a[j+1] = a[j];                         // для втавки элемента
     a[j+1] = t;                               // Вставка элемента
    }
}
```

10.2. Индивидуальные задания

В программе, составленной при выполнении лабораторной работы №9, выполнить сортировку заданным методом по неубыванию массива структур по указанному в индивидуальном задании ключу.

1. Ключ: цена товара. Сортировка *выбором*.
2. Ключ: средний балл. Сортировка *вставкой*.
3. Ключ: время отправления. *Пузырьковая* сортировка.
4. Ключ: количество отработанных за месяц часов. Сортировка *выбором*.
5. Ключ: возраст. Сортировка *вставкой*.
6. Ключ: год издания. *Пузырьковая* сортировка.
7. Ключ: код детали. Сортировка *выбором*.
8. Ключ: дата начала работы. Сортировка *вставкой*.
9. Ключ: суммарный балл. *Пузырьковая* сортировка.
10. Ключ: время вылета. Сортировка *выбором*.
11. Ключ: время отправления. Сортировка *вставкой*.
12. Ключ: средний балл. *Пузырьковая* сортировка.
13. Ключ: дата приемки в ремонт. Сортировка *выбором*.
14. Ключ: номер телефона абонента. Сортировка *вставкой*.
15. Ключ: номер карточки. *Пузырьковая* сортировка.

Библиотека БГУИР

Лабораторная работа №11
ПОИСК ПО КЛЮЧУ В ОДНОМЕРНОМ МАССИВЕ СТРУКТУР

11.1. Поиск в массиве

Поиск – это нахождение информации в массиве по заданному значению (ключу).

11.1.1. Линейный поиск (*метод полного перебора*)

Метод применяется для неупорядоченных массивов и представляет собой последовательный перебор элементов до обнаружения требуемого ключа или до конца массива, если ключ не обнаружен.

```
int P_Lin1(int a[ ], int n, int x)
{
   for(int i = 0; i < n; i++)
       if (a[i] == x) return i;
   return -1;
}
```

Эффективность такого алгоритма пропорциональна количеству элементов.

Единственная возможность улучшить вышеприведенный алгоритм – уменьшить количество проверок на каждом шаге. Для этого вводится вспомогательный элемент – **барьер**, который предохраняет от перехода за пределы массива:

```
int P_lin2(int a[], int  n, int x)
{
  a[n+1] = x;
  int i = 0;
  while (a[i] != x)
                i++;
  if (i == n+1) return -1;
          else return i;
}
```

Эффективность этого алгоритма в два раза выше предыдущего.

11.1.2. Двоичный (*бинарный*) *поиск*

Применяется только для упорядоченных массивов.

Суть метода: выбирается средний элемент и сравнивается с искомым. Если искомый элемент меньше среднего, то из рассмотрения исключается правая половина массива, иначе – левая. Процесс повторяется до тех пор, пока не останется один элемент. Если оставшийся элемент не является искомым, то делается вывод об отсутствии элемента в массиве.

Библиотека БГУИР

```
int P_Dv (int a[], int n, int x)
{
   int i = 0, j = n-1, m;
   while (i<j)
      {
         m = (i+j)/2;            // Вычисление индекса среднего элемента
         if (x > a[m]) i = m+1;  // Исключение левой половины массива
                  else j = m;    // Исключение правой половины массива
      }
   if (a[i] == x) return i;       // Искомый элемент найден
            else return -1;       // Искомый элемент не найден
}
```

11.2. Индивидуальные задания

В программе, составленной при выполнении лабораторной работы №10, *найти в отсортированном массиве структур заданный элемент методами полного перебора и двоичного поиска* (*для упрощения предположить наличие только одного элемента в массиве с заданными характеристиками*).

1. Найти товар ценой *150 000* рублей.
2. Найти студента, имеющего средний балл *7,3*.
3. Найти автобус, отправляющийся в рейс в *13.00*.
4. Найти сотрудника, отработавшего за месяц *156* часов.
5. Найти спортсмена, которому *28* лет.
6. Найти книгу *1966* года издания.
7. Найти деталь с кодом *89383*.
8. Найти сотрудника, работающего с *1975* года.
9. Найти абитуриента, набравшего *287* баллов.
10. Найти самолет, вылетающий в *14.00*.
11. Найти поезд, отправляющийся в *21.00*.
12. Найти студента со средним баллом *8,3*.
13. Найти телевизор, сданный в ремонт *25* числа.
14. Найти абонента с номером *21603*.
15. Найти покупателя с номером карточки *22458*.

Библиотека БГУИР

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. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб. : Питер, 2004.

7. Павловская, Т. А. C++. 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 C++ / Б. Страуструп. – СПб. : БИНОМ, 1999.

Библиотека БГУИР

Св. план 2009, поз. 50

Учебное издание

Навроцкий Анатолий Александрович
Коренская Ирина Николаевна
Бусько Виталий Леонидович

ОСНОВЫ АЛГОРИТМИЗАЦИИ. ПРОГРАММИРОВАНИЕ В СРЕДЕ VISUAL C++

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

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

Подписано в печать 08.10.2009. Формат 60x84 1/16. Бумага офсетная.
Гарнитура «Таймс». Печать ризографическая. Усл. печ. л. 2,91.
Уч.-изд. л. 2,7. Тираж 200 экз. Заказ 191.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6

Библиотека БГУИР