**Л Е К Ц И Я № 05 УКАЗАТЕЛИ**

Оглавление

[1. ПОНЯТИЕ УКАЗАТЕЛЯ 2](#_Toc5206209)

[**Пример 3.1** Взятие адреса переменных разных типов. 3](#_Toc5206210)

[**Пример 3.2** Манипулирование данными с помощью указателей 7](#_Toc5206211)

[**Пример 3.3** Что хранится в указателях 11](#_Toc5206212)

[2. АРИФМЕТИКА УКАЗАТЕЛЕЙ 13](#_Toc5206213)

[**Пример 3.4** Арифметика указателей 14](#_Toc5206214)

[3. ПЕРЕДАЧА ПАРАМЕТРОВ ПО ЗНАЧЕНИЮ И ПО АДРЕСУ 16](#_Toc5206215)

[**Пример 3.5** Передача параметров по значению, ссылке, указателю. 17](#_Toc5206216)

[4. УКАЗАТЕЛИ НА МАССИВ 22](#_Toc5206217)

[**Пример 4.3** Указатели на массив 23](#_Toc5206218)

[**Пример 4.4** Указатели на функцию в качестве параметра функции 28](#_Toc5206219)

[5. УКАЗАТЕЛИ И МАССИВЫ КАК ПАРАМЕТРЫ ФУНКЦИИ 31](#_Toc5206220)

[**Пример 4.5** Передача массива в функцию 33](#_Toc5206221)

[6. МНОГОМЕРНЫЕ МАССИВЫ 39](#_Toc5206222)

[**Пример 4.6** Объявление и инициализация многомерных массивов 40](#_Toc5206223)

[**Пример 4.7** Изменение вектора и матрицы в функции 44](#_Toc5206224)

[**Пример 4.8** Умножение векторов и матриц 49](#_Toc5206225)

[7. ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ 61](#_Toc5206226)

[**Пример 4.9** Динамическое выделение памяти для переменных 63](#_Toc5206227)

[8. ОПЕРАТОР new ДЛЯ СОЗДАНИЯ ДИНАМИЧЕСКИХ МАССИВОВ 66](#_Toc5206228)

[**Пример 4.10** Динамическое выделение памяти для массивов 68](#_Toc5206229)

[**Пример 4.11** Чтение динамических массивов средствами С++ 72](#_Toc5206230)

[**Пример 4.12** Чтение динамических массивов средствами С 81](#_Toc5206231)

[**Пример 4.12** Чтение **статических** массивов 88](#_Toc5206232)

1. ПОНЯТИЕ УКАЗАТЕЛЯ

При сохранении данных компьютерная программа должна знать:

1. Где хранится информация (адрес в памяти)
2. Вид (тип) хранящейся информации (сколько места выделять)
3. Какое значение хранится в памяти

Для реализации этих требований использовался один из возможных методов – объявление и определение переменной. Оператор объявления определяет тип и символическое имя значения, а также заставляет программу выделить область памяти, соответственно типу, для хранения значения и внутренними средствами отслеживать ее адрес. В большинстве случаев программисту необязательно знать реальный адрес каждой переменной – об этом заботится компилятор. К памяти достаточно обратиться с помощью заданного имени.

Существует и второй метод, основанный на указателях, которые являются переменными, хранящими *адреса значений* вместо самих значений.

Но сначала обратимся к способу явного определения адресов переменных. Для этого используется операция взятия адреса переменной - **&**. Если x – переменная, то &x – адрес этой переменной в памяти машины.

Обращаю внимание на то, операция **&** ранее использовалась в битовой операции поразрядного умножения. Это пример *перегрузки* операции, которая заключается в том, что один и тот же знак операции может иметь разные значения. Компилятор определяет назначение знака операции исходя из контекста программы (аналог – омонимы в русском языке, например, климат, хвост, коса).

**Пример 3.1** Взятие адреса переменных разных типов.

В данном случае, используя обыкновенные переменные, можно обработать значение как именованную величину, а его адрес – как производную величину. Адреса переменных растут снизу вверх.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*----------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : Ex03\_01 \*

\* File Name : Ex03\_01.CPP \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modifyed By : \*

\* Created : 13/04/04 \*

\* Last Revision : 15/03/18 \*

\* Comment(s) : ВЗЯТИЕ АДРЕСА ПЕРЕМЕННЫХ РАЗНЫХ ТИПОВ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream> // подключение заголовочного файла

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

char ch1 = 'W';

char ch2 = 'R';

int iVar = 123;

long longVar = 65535; // 4 байта

double dVar = 2.718281828459045; //8 байтов

//разница между адресами равна длине типа - если подряд

cout << " ch1 = \t" << ch1;

cout << " \t Address of ch1 = \t" << &ch1

<< "\t\t" << (int)&ch1 << endl;

cout << " ch2 = \t" << ch2;

cout << " \t Address of ch2 = \t" << &ch2

<< "\t" << (int)&ch2 << endl;

cout << " iVar = \t" << iVar;

cout << " \t Address of iVar = \t" << &iVar

<< "\t" << (int)&iVar << endl;

cout << " longVar = \t" << longVar;

cout << " \t Address of longVar = \t" << &longVar

<< "\t" << (int)&longVar << endl;

cout << " dVar = \t" << dVar;

cout << "\t Address of dVar = \t" << &dVar

<< "\t" << (int)&dVar << endl;

cout << endl;

system("pause");

return 0; //возврат из функции

}// main()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Ex03\_01.CPP file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Рассмотрим ситуацию объявления переменной с точки зрения адреса. В соответствии с ней адрес значения трактуется как именованная величина, а значение – как производная (порожденная) величина. Существует специальный тип переменной – *указатель* *(pointer)*– который хранит **адрес** значения. Имя указателя представляет адрес значения в памяти.

Объявляется переменная типа указатель следующим образом:

int \* pAge; //указатель на целое

int Age; //целое значение

В объявлении указателя обязательно ставится символ «**\***», по которому компилятор узнает, что переменная будет хранить адреса. При **последующем** использовании указателя символ «**\***» перед указателем играет другую роль.

В предыдущем случае, зная значение, можно было найти адрес, где это значение хранится. Сейчас ситуация обратная – зная адрес, надо найти значение, которое по этому адресу хранится. Применяя к указателю операцию «**\***», называемую *косвенным значением* или операцией *разыменования*, получаем значение, хранящееся по данному адресу. Используя указатель, адрес можно обработать как именованную величину, а значение по адресу как производную.

Пусть pAge – указатель на целое, то есть адрес по которому хранится целое число, а \*pAge – целое значение по этому адресу (содержимое адреса). Комбинация \*pAge становится эквивалентом обычной переменной типа int (см. пример 3.2).

Используя обыкновенные переменные, можно обработать значение как именованную величину, а его адрес – как производную величину.

Используя указатели, адрес можно обработать как именованную величину, а значение по этому адресу– как производную величину.

Например,

int \* pAge; //указатель на целое

int N = 20; //целое значение

pAge = &N; //инициализация указателя адресом

В результате операции \*pAge получим значение 20.

От **значения** переход к **адресу** через **&**,

от **адреса** переход к **значению** через **\*** .

**Пример 3.2** Манипулирование данными с помощью указателей

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*---------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\Ex03\_02 \*

\* File Name : Ex03\_02.CPP \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modifyed By : \*

\* Created : 16/04/04 \*

\* Last Revision : 17/03/18 \*

\* Comment(s) : МАНИПУЛИРОВАНИЕ ДАННЫМИ С ПОМОЩЬЮ УКАЗАТЕЛЕЙ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

//setlocale(LC\_ALL, "C"); //отключение русского языка

system("color F0"); //экран белый, буквы черные

system("cls"); //очистка экрана

//----- П Е Р Е М Е Н Н Ы Е ------

int myAge; //переменная

int \* pAge = 0; //указатель

cout << "\tEx03\_02.cpp\n";

//разница между адресами равна длине типа

cout << endl;

cout << " Установим значение myAge = 5. . .\t" << endl;

myAge = 5;

cout << " myAge = \t" << myAge << endl; //начальное значение

pAge = &myAge; //в указатель pAge положить адрес myAge

cout << " \*pAge = \t" << \*pAge; //начальное значение через укзатель

cout << "\n\n";

cout << " Установим значение \*pAge = 7 . . .\n";

\*pAge = 7; //по адресу из указателя myAge присвоить 7

cout << " myAge = \t" << myAge << endl; //новое значение

cout << " \*pAge = \t" << \*pAge << endl; //новое значение через укзатель

cout << endl;

cout << " Установим значение myAge = 20 . . .\n"; myAge = 20;

cout << " myAge = \t" << myAge << endl; //новое значение

cout << " \*pAge = \t" << \*pAge << endl;//новое значение через указатель

cout << endl;

cout << " &myAge = " << &myAge

<< "\t pAge = " << pAge //работа с одним и тем же адресом

<< "\t &pAge = " << &pAge << "\n\n"; //работа с одним и тем же адресом

double dVar = 2.718281828459045;

double \* pDoub;

pDoub = &dVar;

cout << " dVar = " << dVar << " &dVar = " << &dVar

<< " sizeof dVar = " << sizeof dVar << "\n";

cout << " \*pDoub = " << \*pDoub << " pDoub = " << pDoub

<< " sizeof pDoub = " << sizeof pDoub

<< " &pDoub = " << &pDoub << "\n";

/\*

int \*p; //проблема с неинициализированной переменной

\*p = 12345;

cout << p << " " << \*p << " " << &p << endl;

\*/

cout << endl;

system("pause");

return 0;

} //main()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Ex03\_02.CPP file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Переменная myAge типа int и переменная-указатель pAge на int – это две стороны одной и той же переменной. Переменная myAge представляет прежде всего значение, а операция «**&»** используется для получения адреса этой переменной. Переменная pAge – представляет прежде всего адрес, а операция «**\***» служит для получения значения по этому адресу. Поэтому \*pAge и myAge синонимы и выражение \*pAge можно использовать точно также как и переменную типа int.

Переменная-указатель почти никогда не бывает просто указателем (void). Она всегда указывает на определенный тип данных. Тип указателя сообщает компилятору сколько памяти необходимо для объекта, адрес которого указатель содержит. Например, адрес переменной типа int выглядит также как и адрес переменной типа double, но для типов int и double требуется различное количество байтов и различные внутренние форматы для хранения значений. Сами переменные-указатели имеют одинаковый размер (обычно 4 байта).

При объявлении указателей надо быть внимательным:

int \*p1, p2;

и

int \*p1, \*p2; – это разные объявления для p2.

В первом случае объявляются указатель на целое и целое, во втором – два указателя на целое.

Указатели, значения которых равны 0, называются *пустыми* (null pointer). Если заранее неизвестно, какой адрес должен храниться в указателе, ему лучше присвоить значение 0. Неинициализированные указатели называются *дикими*. Они очень опасны!

int \*x;

\* = 223323; //это может быть присваивание по адресу ОС(!), так как адрес в указателе x берется произвольно

int x, \*y; //**пример неинициализированных переменных**

cout <<*"x = " << x << "\t y = " << y << “\t” << (int)y*

<< "\t\*y = " << \*y <<endl;//**системная ошибка !!**

Оставить только

cout <<*"x = " << x << "\t y = " << y << “\t” << (int)y* <<endl;

*y = 0xCCCCCCCC* // **1**100 1100 1100. . . – отрицательное число

(int)y выдаст отрицательное число, которое не может быть адресом и обращение к отрицательному адресу некорректно.

Указатели позволяют манипулировать переменными, не обращая внимания на их реальные значения.

**Пример 3.3** Что хранится в указателях

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*---------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\Ex03\_03 \*

\* File Name : Ex03\_03.CPP \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modifyed By : \*

\* Created : 16/04/04 \*

\* Last Revision : 17/03/18 \*

\* Comment(s) : ЧТО ХРАНИТСЯ В УКАЗАТЕЛЯХ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

//setlocale(LC\_ALL, "C"); //отключение русского языка

system("color F0"); //экран белый, буквы черные

system("cls"); //очистка экрана

//----- П Е Р Е М Е Н Н Ы Е ------

int myAge = 5; //переменная

int yourAge = 10;

int \* pAge = &myAge; //укзатель инициализирован адресом myAge

cout << "\tEx03\_03.cpp\n";

cout << endl;

cout << " Присвоим : pAge = &myAge . . .\n\n";

**// 1 BEGIN**

cout << " myAge = \t" << myAge; //начальные значения

cout << "\t\t yourAge = \t" << yourAge << endl;

// cout << endl;

cout << " Адрес : &myAge = \t" << &myAge; //адреса

cout << "\t &yourAge = \t" << &yourAge << endl;

// cout << endl;

cout << " pAge = \t" << pAge << endl; //указатели

cout << " \*pAge = \t" << \*pAge << endl;

cout << "\t &pAge = \t" << &pAge << "\n\n\n"; //адрес самого указателя

**// 1 END**

cout << " ПереПрисвоим : pAge = &yourAge . . .\n

pAge = &yourAge; //укзатель инициализирован адресом yourAge

//скопировать предыдущую часть 1 BEGIN - 1 END

cout << " myAge = \t" << myAge; //начальные значения

cout << "\t\t yourAge = \t" << yourAge << endl;

// cout << endl;

cout << " Адрес : &myAge = \t" << &myAge; //адреса

cout << "\t &yourAge = \t" << &yourAge << endl;

cout << "\t\t\t\t\t pAge = \t" << pAge << endl; //указатели

cout << "\t\t\t\t\t \*pAge = \t" << \*pAge << endl;

cout << "\t &pAge = \t" << &pAge << endl; //адрес самого указателя

cout << endl;

cout << endl;

system("pause");

return 0;

} //end main()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Ex03\_03.CPP file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. АРИФМЕТИКА УКАЗАТЕЛЕЙ

К указателям (кроме указателей на тип void) могут применяться арифметические операции. Компилятор, зная тип указателя, вычисляет размер переменной этого же типа, после чего модифицирует адрес, содержащийся в указателе, в соответствии с заданной арифметической операцией, но с учетом вычисленного для данного типа размера. Это значит, что для указателя на тип double операция инкремента увеличит значение адреса не на 1, а на 8 байтов в соответствии с длиной типа double.

double \*pd = 0;

pd++;

**Пример 3.4** Арифметика указателей

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*---------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\Ex03\_03 \*

\* File Name : Ex03\_04.CPP \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modifyed By : \*

\* Created : 16/04/04 \*

\* Last Revision : 17/03/18 \*

\* Comment(s) : АРИФМЕТИКА УКАЗАТЕЛЕЙ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

//setlocale(LC\_ALL, "C"); //отключение русского языка

system("color F0"); //экран белый, буквы черные

system("cls"); //очистка экрана

cout << "\tEx03\_04.cpp\n";

double dNumber;

double \*dptr = &dNumber;

double \*dptr0; //для хранения начального адреса

dptr0 = dptr; //сохраним начальное значение указателя

cout << " Начало : dptr = " << dptr << " " << (int)dptr << endl;

dptr++; //показать на соседний элемент double

cout << " Инкремент : ++dptr = " << dptr << " " << (int)dptr << endl;

cout << " Число элементов = " << dptr - dptr0 << endl;

cout << endl;

dptr++; //показать на соседний элемент double

cout << " Инкремент : ++dptr = " << dptr << " " << (int)dptr << endl;

cout << " Число элементов = " << dptr - dptr0 << endl << endl;

cout << " dptr0 = " << dptr0 << "\tdptr0 + 2 = " << dptr0 + 2 << endl;

cout << " \t\t\t dptr = " << dptr << endl;

cout << endl;

system("pause");

return 0;

} //end main()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Ex03\_04.CPP file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. ПЕРЕДАЧА ПАРАМЕТРОВ ПО ЗНАЧЕНИЮ И ПО АДРЕСУ

Существуют два способа передачи параметров в функцию – по *значению* и по *адресу*.

При передаче параметров *по значению* компилятор создает временную копию объекта, который должен быть передан и размещает ее в области памяти, выделяемой при вызове функции (в стеке). Вызываемая функция оперирует именно с этой копией, не оказывая на оригинал никакого воздействия (разные адреса хранения). Поэтому внутри функции (в копии) параметры будут изменены, а у вызывающей функции останется неизмененный оригинал. Доступа к исходным значениям параметров у функции нет и нет возможности их изменить.

При передаче параметров *по адресу* вызываемой функции передается адрес фактического параметра вызывающей функции и вызываемая функция вносит изменения по этому адресу. В стек заносятся копии адресов аргументов, а функция осуществляет доступ к ячейкам памяти по этим адресам. Поэтому, когда вызываемая функция отработала и освободила выделенную ей память, изменения, сделанные ею по адресам фактических параметров, остались у вызывающей функции. Исходные значения аргументов изменились.

Передача параметров по адресу имеет две разновидности – по ссылке и по указателю.

*Что есть ссылка?*  Ссылка – особый тип данных, являющийся скрытой формой указателя, который при использовании автоматически разыменовывается. Ссылка – это разыменованный константный указатель, то есть по сути своей адрес, но уже разыменованный, причем адрес менять нельзя, а значение по адресу можно. В функцию передается адрес параметра, внутри функции все обращения к параметру **неявно** разыменовываются.

При передаче параметра в функцию по *адресу* с помощью *указателя* используется операция взятия адреса &, а для получения его значения в функции требуется операция разыменовывания.

**Пример 3.5** Передача параметров по значению, ссылке, указателю.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*---------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\Ex03\_03 \*

\* File Name : Ex03\_05.CPP \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modifyed By : \*

\* Created : 16/04/04 \*

\* Last Revision : 17/03/18 \*

\* Comment(s) : ПЕРЕДАЧА ПАРАМЕТРОВ ПО ЗНАЧЕНИЮ. ИЗМЕНЕНИЙ НЕТ \*

\* ПЕРЕДАЧА ПАРАМЕТРОВ ПО ССЫЛКЕ. ИЗМЕНЕНИЯ ЕСТЬ \*

\* ПЕРЕДАЧА ПАРАМЕТРОВ ПО УКАЗАТЕЛЮ. ИЗМЕНЕНИЯ ЕСТЬ \*

\* ПЕРЕДАЧА ПАРАМЕТРОВ ПО УКАЗАТЕЛЮ БЕЗ ПРАВА ИЗМЕНЕНИЯ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Г Л О Б А Л Ь Н Ы Е П Е Р Е М Е Н Н Ы Е \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int globVar = 4;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* З А Г О Л О В К И Ф У Н К Ц И Й \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// **int x** и **int &x** - две стороны одной и той же переменной:

// **имя** и **разыменованный адрес**

void f1\_Val(int x, int y); //передаются значения

void f2\_Ref(int &x, int &y); //передаются ссылки

void f3\_Ptr(int \*x, int \*y); //передаются указатели

//передаются указатели без права модификации

void f4\_Ptr\_No(const int \*x, const int \*y);//**без права изменения**

/\*

//перестановка двух чисел ***САМОСТОЯТЕЛЬНО***

void swap1\_Val(int x, int y); //передаются значения

void swap2\_Ref(int &x, int &y); //передаются ссылки

void swap3\_Ptr(int \*x, int \*y); //передаются указатели

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

//setlocale(LC\_ALL, "C"); //отключение русского языка

system("color F0"); //экран белый, буквы черные

system("cls"); //очистка экрана

cout << "\tEx03\_05.cpp\n";

//1

int a = 1, b = 2;

// " ЗНАЧЕНИЕ"

cout << "\n/1\t VALUE \n"; //**изменений нет**

cout << " /1 main: \ta = " << a << "\t b = " << b

<< " \t\t globVar = " << globVar << endl;

f1\_Val(a, b); // Передача данных по **значению**

cout << " /3 main: \ta = " << a << "\t b = " << b

<< " \t\t globVar = " << globVar << endl;

// " ССЫЛКА "

cout << "\n/2\t REFERENCE \n"; //**изменения есть**

cout << " /4 main: \ta = " << a << "\t b = " << b

<< " \t\t globVar = " << globVar << endl;

f2\_Ref(a, b); // Передача данных по **ссылке**

cout << " /6 main: \ta = " << a << "\t b = " << b

<< "\t\t globVar = " << globVar << endl;

// " УКАЗАТЕЛЬ"

cout << "\n/3\t POINTER " << endl; //**изменения есть**

cout << " /7 main: \ta = " << a << "\t b = " << b

<< " \t globVar = " << globVar << endl;

f3\_Ptr(&a, &b); // Передача данных по **указателю**

cout << " /9 main: \ta = " << a << "\t b = " << b

<< " \t globVar = " << globVar << "\n\n";

//передаются указатели **без права модификации**

// f4\_Ptr\_No (&a, &b); // сообщение об **ошибке** !!!

*/\**

*//2* ***САМОСТОЯТЕЛЬНО***

*int x = 5, y = 10;*

*cout << "/2" << endl;*

*cout <<" /main: Before swap \tx = " << x <<"\t y = "<< y << endl;*

*swap1\_Val (x, y); // Передача данных по значению*

*cout <<" /main: After swap1\_Val \tx = " << x <<"\t y = "<< y << endl;*

*swap2\_Ref (x, y); //Передача данных по ссылке*

*cout <<" /main: After swap2\_Ref \tx = " << x <<"\t y = "<< y << endl;*

*swap3\_Ptr (&x, &y); // Передача данных по указателю*

*cout <<" /main: After swap3\_Ptr \tx = " << x <<"\t y = "<< y << endl;*

*\*/*

cout << endl;

system("pause");

return 0;

} //end main()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*-------------------------------------------------------------\*/

/\* Передача данных \*/

/\* по значению \*/

/\*-----------------\*/

void f1\_Val(int x, int y)

{

x = x + 10; //изменение **копии**

y = y + 10; //изменение **копии**

globVar = globVar + 10; //изменение глобальной переменной

cout << " /2 f1\_Val: \tx = " << x << "\t y = " << y

<< "\t\t globVar = " << globVar << endl;

}

/\*-------------------------------------------------------------\*/

/\* Передача данных \*/

/\* по ссылке \*/

/\*-----------------\*/

void f2\_Ref(int & x, int & y)

{

x = x + 10; //изменение **оригинала**

y = y + 10; //изменение **оригинала**

globVar = globVar + 10; //изменение глобальной переменной

cout << " /5 f2\_Ref: \tx = " << x << "\t y = " << y

<< "\t\t globVar = " << globVar << endl;

}

/\*-------------------------------------------------------------\*/

/\* Передача данных \*/

/\* по указателю \*/

/\*-----------------\*/

void f3\_Ptr(int \*x, int \*y)

{

\*x = \*x + 10; //изменение **оригинала**

\*y = \*y + 10; //изменение **оригинала**

globVar = globVar + 10; //изменение глобальной переменной

cout << " /8 f3\_Ptr: \*x = " << \*x << "\t \*y = " << \*y

<< "\t globVar = " << globVar << endl;

}

/\*

/\*-------------------------------------------------------------\*/

/\* Передача данных по указателю \*/

/\* на константные значения \*/

/\*------------------------------\*/

**/\***

//передаются указатели на константные значения

void f4\_Ptr\_No( const int \*x, const int \*y)

{

\*x = \*x + 10; //изменение оригинала **ОШИБКА** !!

\*y = \*y + 10; //изменение оригинала **ОШИБКА** !!

globVar = globVar + 10; //изменение глобальной переменной

cout <<" /10 f4\_Ptr\_No: \t\*x = " << \*x

<<"\t \*y = "<< \*y <<"\t globVar = " << globVar << endl;

}

**\*/**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// ***САМОСТОЯТЕЛЬНО***

//Перестановка двух чисел

**void swap1\_Val (int x, int y)**

{

int temp;

temp = x;

x = y;

y = temp;

}// **swap1\_Val** ()

**void swap2\_Ref (int &x, int &y)**

{

int temp;

temp = x;

x = y;

y = temp;

}// **swap2\_Ref** ()

**void swap3\_Ptr (int \*x, int \*y)**

{

int temp;

temp = \*x;

\*x = \*y;

\*y = temp;

}// **swap3\_Ptr** ()

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Ex03\_05.CPP file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Если необходимо, чтобы функция модифицировала оригинал объекта, используется передача параметров *по адресу*. Эффективность передачи адреса объекта вместо самого объекта проявляется в экономии оперативной памяти (не надо хранить еще одну копию) и в увеличении скорости работы, особенно, если используются большие объекты (не надо тратить время на их копирование, НО может быть несанкционированное изменение данных).

Функции могут возвращать только одно значение. Если требуется, чтобы функция возвратила несколько значений, например, два, то необходимо передать функции *по адресу* два объекта. В ходе выполнения функция присвоит этим объектам новые значения. Передача *по адресу*, позволяющая функции менять исходные объекты, равносильна разрешению функции возвратить два значения. В этом случае можно вовсе обойтись без возвращаемого значения (void) или использовать его для сообщения об ошибках.

При передаче больших структур данных, во избежание их копирования в функции, их можно передавать *по адресу*, но указав const перед всеми параметрами, чтобы исключить их непреднамеренное изменение в функции.

Для передачи параметра *по указателю* без права изменения параметра используется модификатор const. Любая попытка модифицировать такой объект в пределах вызываемой функции вызовет сообщение компилятора об ошибке. Например,

void f4\_Ptr\_No (const int \*x, const int \*y) .

Здесь указатель указывает на константу const int \*x . Указываемое значение есть const целого типа.

1. УКАЗАТЕЛИ НА МАССИВ

Указатели применяют адреса так, как это делается в самой машине, тем самым повышая эффективность программ. Объявление массива, например,

int **myArr**[3]; **- адресная константа**

представляет собой скрытую форму использования указателя, т.к. имя массива является адресом его начала и myArr == &myArr[0] – обе части равенства определяют адрес нулевого элемента. Оба обозначения являются *константами* типа указатель, поскольку они не меняются (**менять их нельзя !**) на протяжении всей программы. Однако их можно присваивать переменной типа указатель и изменять значение этой переменной.

int AVec[3] = {1,2,3};

int BVec[3];

**BVec = Avec; //НЕЛЬЗЯ ! ! !**

**Пример 4.3** Указатели на массив

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*-----------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : Ex04\_03 \*

\* File Name : Ex04\_03\_УКАЗАТЕЛИ НА МАССИВ.CPP \*

\* Programmer(s) : Чечиков Ю.Б. & Безродных И.П. \*

\* Modifyed By : \*

\* Created : 19/04/04 \*

\* Last Revision : 10/07/05 \*

\* Comment(s) : УКАЗАТЕЛИ НА МАССИВ. АРИФМЕТИКА УКАЗАТЕЛЕЙ \*

\*------------------------------------------------------------------------\*

\* Результат работы программы \*

\* /1 \*

\* Int Double \*

\* POINTER 4 Bytes 8 Bytes \*

\* POINTER + 0 0000008A394FF5F8 0000008A394FF668 \*

\* POINTER + 1 0000008A394FF5FC 0000008A394FF670 \*

\* POINTER + 2 0000008A394FF600 0000008A394FF678 \*

\* POINTER + 3 0000008A394FF604 0000008A394FF680 \*

\* \*

\* /2 \*

\* Int Double \*

\* POINTER 4 Bytes 8 Bytes \*

\* POINTER + 0 0000008A394FF5F8 -11 0000008A394FF668 -5.5 -5.5 \*

\* POINTER + 1 0000008A394FF5FC -13 0000008A394FF670 -6.5 -6.5 \*

\* POINTER + 2 0000008A394FF600 -15 0000008A394FF678 -7.5 -7.5 \*

\* POINTER + 3 0000008A394FF604 -17 0000008A394FF680 -8.5 -8.5 \*

\* \*

\* /3 \*

\* \*intArr = -11 \*intArr + 2 = -9 \*

\* \*intArr = -11 (\*intArr) + 2 = -9 \*

\* \*(intArr + 2) = -15 \*

\* intArr[2] = -15 \*

\* \*

\* intArr = 0000008A394FF5F8 \*

\* intArr + 2 = 0000008A394FF600 \*(intArr + 2) = -15 \*

\* intArr [2] = -15 \*

\* \*

\* ВАРИАНТЫ ОБРАЩЕНИЯ К ЭЛЕМЕНТАМ МАССИВА \*

\* intArr[2]= -15 \*(pint + 2)= -15 \*(2 + pint )= -15 2[intArr]= -15 \*

\* intArr[2]= -15 \*(intArr + 2)= -15 \*(2 + intArr)= -15 2[intArr]= -15 \*

\* pint[2] = -15 \*(pint + 2)= -15 \*(2 + pint )= -15 2[pint] = -15 \*

\* -15 -15 -15 -15 \*

\* Для продолжения нажмите любую клавишу . . . \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream> // подключение заголовочного файла

using namespace std;

int main() // описание главной функции

{

setlocale(LC\_ALL, "RUSSIAN");

**//1 укзатель на массив; прибавление к указателю**

cout << "/1\n";

int intArr[4], \*pint; //целый массив и укзатель на целое

int i; //индексная переменная

double dbArr[4], \*pdb; //веществ массив и указатель на веществ

/\* обозначение массива представляет собой скрытую форму указателя

**intArr == &intArr[0]** - это определение адреса нулевого

элемента массива

\*/

pint = intArr; //инициализация указателя адресом начала массива

pdb = &dbArr[0]; //адрес начала массива - это постоянный указатель

//печать адресов элементов массивов

cout << "\t\t" << " Int" << "\t\t Double \n";

cout << " POINTER \t" << " 4 Bytes" << "\t 8 Bytes \n";

for (i = 0; i < 4; i++)

{

cout << " POINTER + " << i << "\t"

<< pint + i << "\t" << pdb + i << "\n";

}//for i

cout << endl;

**//2 движение по массиву через указатель; печать массива**

cout << "/2\n";

// **pint+i == &intArr[i]**; это один и тот же адрес !!!

// **\*(pint+i) == intArr[i];** это одно и то же значение !!!

//заполнение массива,

//печать адресов элементов и значений элементов массивов

cout << "\t\t" << " Int" << "\t\t\t Double \n";

cout << " POINTER \t" << " 4 Bytes" << "\t\t 8 Bytes \n";

//заполняем массив и печатаем его

for (i = 0; i < 4; i++)

{

\*(pint + i) = -(2 \* i + 11); //нечетное число в целом массиве

**//сначала без точки - не 11.0, а 11 - целый результат**

dbArr[i] = -(2 \* i + 11.0) / 2; //дробное число в веществ массиве

cout << "POINTER + " << i << "\t"

<< pint + i << " " << \*(pint + i) << "\t"

<< pdb + i << " " << \*(pdb + i) << " "

<< dbArr[i] << "\n";

}//for i

cout << endl;

// адреса и значения

cout << "/3\n";

//\*intArr + 2 операция \* имеет более высокий приоритет, чем +

// поэтому \*intArr + 2 == (\*intArr) + 2

//не путать с \*(intArr + 2)

//меняем содержимое 0-го элемента - имя массива

cout << " \*intArr = " << \*intArr

<< " \*intArr + 2 = " << \*intArr + 2 << endl;

//результаты совпадут

cout << " \*intArr = " << \*intArr

<< " (\*intArr) + 2 = " << (\*intArr) + 2 << endl;

cout << " \*(intArr + 2) = " << \*(intArr + 2) << "\n";

cout << " intArr[2] = " << intArr[2] << "\n\n";

//доступ к 3-му элементу массива

cout << " intArr = " << intArr << "\n";

cout << " intArr + 2 = " << intArr + 2

<< " \*(intArr + 2) = " << \*(intArr + 2) << endl;

cout << "\t\t\t\t intArr [2] = " << intArr[2] << "\n\n";

**//ВАРИАНТЫ ОБРАЩЕНИЯ К ЭЛЕМЕНТАМ МАССИВА !!!**

**cout << " ВАРИАНТЫ ОБРАЩЕНИЯ К ЭЛЕМЕНТАМ МАССИВА\n";**

**cout << " intArr[2] = " << intArr[2] //через имя массива**

**<< " \*(pint + 2) = " << \*(pint + 2) //через указатель на начало массива**

**<< " \*(2 + pint ) = " << \*(2 + pint) //через указатель на начало массива**

**<< " 2[intArr] = " << 2[intArr] << endl; //через имя массива**

**cout << " intArr[2] = " << intArr[2] //через имя массива**

**<< " \*(intArr + 2) = " << \*(intArr + 2) //через имя массива**

**<< " \*(2 + intArr) = " << \*(2 + intArr) //через имя массива**

**<< " 2[intArr] = " << 2[intArr] << endl; //через имя массива**

**cout << " pint[2] = " << pint[2] //через указатель на начало массива**

**<< " \*(pint + 2) = " << \*(pint + 2) //через указатель на начало массива**

**<< " \*(2 + pint ) = " << \*(2 + pint) //через указатель на начало массива**

**<< " 2[pint] = " << 2[pint] << endl; //через указатель на начало массива**

**cout << "\t" << intArr[2] << " " << \*(pint + 2) << " " << \*(2 + pint) << " " << 2[intArr] << endl;**

system("PAUSE");

return 0;

}

/\*\*\*\*\*\*\*\* End of Ex03\_05\_ПЕРЕДАЧА ПАРАМЕТРОВ.CPP file \*\*\*\*\*\*/

Что происходит, когда мы прибавляем единицу к указателю? Компилятор добавит *единицу памяти*, равную размеру одного элемента, то есть мы перейдем к адресу следующего *элемента* массива, а не байта. Вот почему мы должны специально оговаривать тип объекта, на который ссылается указатель и одного адреса здесь недостаточно. **Арифметика указателей на данные любого типа выполняется относительно объекта, на который указывает указатель.**

**Пример 4.4** Указатели на функцию в качестве параметра функции

(НЕОБЯЗЯТЕЛЬНО)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*-------------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\Lesson\_03\_Указатели\_Ссылки\Ex04\_04 \*

\* Project Type : Win32 Console Application \*

\* File Name : Ex04\_04.CPP \*

\* Programmer(s) : Чечиков Ю.Б. & Безродных И.П. \*

\* Modifyed By : \*

\* Created : 09/06/05 \*

\* Last Revision : 04/09/06 \*

\* Comment(s) : УКАЗАТЕЛЬ НА ФУНКЦИЮ В КАЧЕСТВЕ ПАРАМЕТРА ФУНКЦИИ \*

\* РАССЧЕТ ИНТЕГРАЛА ДЛЯ ФУНКЦИЙ \*

\* F1 = X \*

\* F2 = -X \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <math.h> //подключение математических функций fabs()

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//вычисление интеграла задаваемой функции

float integ (float a, float b, float eps, float (\*f)(float x));

// задаваемые функции

float f1 (float x); //F1 = X

float f2 (float x); //F2 = -X

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* О С Н О В Н А Я П Р О Г Р А М М А \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main ()

{

float a,b,eps;

float i1, i2;

//ввод данных и эхо-печать

cout << "Enter A, B, Eps ";

cin >> a >> b >> eps;

cout <<" a = " << a << " b = " << b <<" eps = " << eps << "\n\n";

//вычисление интегралов для разных функций

//имя функции - это адрес ее начала

i1 = integ (a,b,eps,f1);

i2 = integ (a,b,eps,f2);

//печать результатов

cout << " Integral1 = " << i1 << "\n";

cout << " Integral2 = " << i2 << "\n";

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//\*-------------------------------------------------------------------\*

//\* рассчет интеграла с \*

//\* заданной точностью \*

//\* методом прямоугольников \*

//\*--------------------------\*

float integ (float A, float B, float Eps, float (\*F)(float x))

{

float x; //аргумент

int N; //число разбиений интервала [A,B]

float h; //шаг приращения аргумента

float S\_New; //текущее значение интеграла (Сумма Новая)

float S\_Old; //предыдущее значение интеграла (Сумма Старая)

float diff; //разность между текущим и предыдущим значениями интеграла

int i; //переменная цикла

//инициализация переменных

N = 1; //число разбиений - весь интервал [A,B]

S\_Old = 0.0;

S\_New = 0.0;

diff = 2 \* Eps; //для входа в цикл

//рассчет интеграла с заданной точностью

while (diff > Eps) //пока разница между суммами больше Eps

{

N++; //увеличение числа разбиений отрезка

h = (B - A) / N; //новая величина приращения аргумента

S\_New = 0.0; //новый интеграл пока не посчитали

//cout << "\n N = " << N << "\th = " << h << endl;

for (i = 0; i <= N-1; i++) //вычисление площади интеграла

{

x = A + i \* h; //приращение аргумента

S\_New += F(x) \* h; //приращение интеграла

//cout << " x = " << x << "\tS\_New = " << S\_New << endl;

}//for

//разность текущего и предыдущего значений интеграла

diff = fabs(S\_New - S\_Old);

S\_Old = S\_New; //для новой итерации

}//while

return S\_New; //рассчитанный интеграл

} //integ()

//\*-------------------------------------------------------------------\*

//\* заданная функция1 \*

//\*--------------------\*

float f1 (float x) //F1 = X

{

return x;

}

//\*-------------------------------------------------------------------\*

//\* заданная функция2 \*

//\*--------------------\*

float f2 (float x) //F2 = -X

{

return -x;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of Ex04\_04.CPP file \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. УКАЗАТЕЛИ И МАССИВЫ КАК ПАРАМЕТРЫ ФУНКЦИИ

Массивы могут быть аргументами функции. Проанализируем скелет программы.

int main()

{

int ages[50]; //массив на 50 элементов

. . .

convert (ages);

. . .

}

int convert (int years[])

{

//каков размер массива?

. . .

}

Очевидно, что массив ages состоит из 50 элементов. А что можно сказать про массив years ? В программе нет такого массива. Описатель int years[] задает не массив, а *указатель*  на него и никакой информации о размере массива нет. Рассмотрим вызов функции.

convert (ages);

Здесь ages – аргумент функции. Имя массива ages является *указателем* на начало массива (нулевой элемент). Таким образом, оператор вызова функции передает указатель, то есть адрес массива ages. Это значит, что аргумент функции является указателем и мы можем написать функцию convert() следующим образом:

int convert (int \*years)

{

. . .

**\*(years + 2)** - можно

}

Действительно, операторы

int years[]; //аналогично char S[] = “ASDF”;

int \*years;

синонимы. Оба объявляют переменную years указателем на массив целых чисел. Но только первый из них напоминает нам, что указатель years ссылается на массив. (Как теперь связать его с массивом ages?) Дело в том, что При использовании указателя в качестве аргумента (см. передачу параметров по адресу), функция работает с переменной в вызывающей программе. Поэтому операторы функции convert(), использующие указатель years, фактически работают с массивом ages, находящимся в функции main().

Когда имя массива применяется в качестве аргумента, то функции передается указатель, пользуясь которым, функция изменяет полученный массив в основной программе. **Функция работает с массивом, находящимся в другой части программы.** В функции локального выделения памяти под массив не происходит.

Поскольку при передаче массива функции передается адрес начала массива, то возникает вопрос: «Где заканчивается массив? Как узнать, что мы не вышли за границу массива?». Со строками все просто – нулевой символ (‘\0’) маркирует конец строки, но что делать с массивом чисел, для которого нет признака конца массива? В этом случае приходится передавать в функцию еще один параметр – количество элементов массива.

**Пример 4.5** Передача массива в функцию

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*---------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : Ex04\_05 \*

\* File Name : Ex04\_05 ПЕРЕДАЧА МАССИВА.CPP \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modifyed By : \*

\* Created : 16/04/04 \*

\* Last Revision : 12/03/19 \*

\* Comment(s) : ИСПОЛЬЗОВАНИЕ УКАЗАТЕЛЕЙ ПРИ ПЕРЕДАЧЕ МАССИВОВ\*

\* ИЗМЕНЕНИЕ МАССИВА В ФУНКЦИИ \*

\* ОСОБЕННОСТИ ВЫЧИСЛЕНИЯ \*

\* sizeof(globVec) / sizeof(int) \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Г Л О Б А Л Ь Н Ы Е П Е Р Е М Е Н Н Ы Е \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int globVec[10]; //для проверки sizeof(globVec)/sizeof(int)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* З А Г О Л О В К И Ф У Н К Ц И Й \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double Average1(int Vec[], int sizeVec); //через массив

double Average2(int \*pVec, int sizeVec);//через указатель на массив

double Average3(int &rVec, int sizeVec); //через ссылку на массив

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

const int NMAX = 8; //максимальный размер массива

int AVec[NMAX]; //целый массив

int i; //индексная переменная

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

//setlocale(LC\_ALL, "C"); //отключение русского языка

system("color F0"); //экран белый, буквы черные

system("cls"); //очистка экрана

cout << "\t Ex04\_05 ПЕРЕДАЧА МАССИВА.CPP \n";

//1 передается массив

**cout << "/1\tПЕРЕДАЕТСЯ ИМЯ МАССИВА\n";**

//заполнение массива

cout << "\t Исходный Массив \n";

for (i = 0; i < NMAX; i++)

{

AVec[i] = i + 1; //целые числа от 1 до NMAX

cout << "\t/main : AVec[" << i << "]= " << AVec[i]

<< "\t" << &AVec[i] << endl;

}//for i

cout << "\n\n";

**//Операция sizeof(AVec)/sizeof(int)работает правильно ,**

**// так как здесь мы имеем дело с массивом**

cout << " /main sizeof : " << sizeof(AVec)

<< "/" << sizeof(int) << " = "

<< sizeof(AVec) / sizeof(int) << "\n\n";

//печать среднего значения

cout << "\t/main : Average1 = "

<< Average1(AVec, NMAX) << endl;

//массив после изменения

cout << "\n\tНовый AVec \n";

for (i = 0; i < NMAX; i++)

{

cout << "\t/main : AVec[" << i << "]= "

<< AVec[i] << endl;;

}//for i

cout << " \n";

//2 передается указатель на массив

**cout << "\n/2\tПЕРЕДАЕТСЯ УКАЗАТЕЛЬ НА МАССИВ\n";**

//печать среднего значения

cout << "\t/main : Average2 = "

<< Average2(AVec, NMAX) << endl;

//3 передается ссылка на массив

**cout << "\n/3\tПЕРЕДАЕТСЯ ССЫЛКА НА МАССИВ\n";**

//печать среднего значения

//синонимы

//\*AVec - разыменованный адрес целого значения

cout << "\t/main : Average3(\*AVec, NMAX) = "

<< Average3(\*AVec, NMAX) << "\n\n";

cout << "\t/main : Average3(AVec[0], NMAX) = "

<< Average3(AVec[0], NMAX) << "\n\n";

cout << endl;

system("pause");

return 0;

} //end main()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*-------------------------------------------------------------\*/

/\* Подсчет среднего значения \*/

/\* целочисленного массива \*/

/\*----------------------------\*/

**double Average1 ( int Vec[] //имя массива**

**, int sizeVec) //длина массива**

{

int Sum = 0;

int i;

// операчия sizeof(Vec)/sizeof(int) работает неправильно,

// так как Vec - указатель на int, имеющий длину 4 байта

// чтобы операция сработала - массив должен быть глобальным !!!

//размер указателя

cout << "/Aver1 : " << sizeof(Vec)

<< "/" << sizeof(int) << " = "

<< sizeof(Vec) / sizeof(int) << endl;

//размер массива

cout << "/Aver1 Glob : " << sizeof(globVec)

<< "/" << sizeof(int) << " = "

<< sizeof(globVec) / sizeof(int) << endl;

//подсчет среднего

for (i = 0; i < sizeVec; i++)

Sum = Sum + Vec[i];

cout << "/Aver1 : Sum = " << Sum

<< " sizeVec = " << sizeVec << endl;

//изменим массив в процедуре - результаты видны в программе

for (i = 0; i < sizeVec; i++)

{

Vec[i] = i \* 2; //четные числа

cout << "\t/Aver1 : Vec[" << i << "]= " << Vec[i]

<< "\t" << &Vec[i] << endl;

}//for i

cout << " \n";

return 1.0 \* Sum / sizeVec; //для преобразования к double

// нельзя sum/sizeVec \* 1.0

}//Average1()

/\*-------------------------------------------------------------\*/

/\* Подсчет среднего значения \*/

/\* целочисленного массива \*/

/\* через указатели \*/

/\*----------------------------\*/

**double Average2 ( int \* pVec //указатель**

**, int sizeVec) //длина массива**

{

double Sum = 0; //поменяли тип на удобный

int i;

for (i = 0; i < sizeVec; i++)

{

Sum = Sum + \*(pVec + i);

cout << "\t/Aver2 : \*(pVec +" << i << ")= " << \*(pVec + i)

<< "\t" << (pVec + i) << endl;

}//for i

cout << endl;

return Sum / sizeVec; //БЕЗ преобразования к double

//return 1.0 \* Sum / sizeVec; //для преобразования к double

// нельзя sum/sizeVec \* 1.0

}//Average2()

/\*-------------------------------------------------------------\*/

/\* Подсчет среднего значения \*/

/\* целочисленного массива \*/

/\* через ссылки \*/

/\*----------------------------\*/

**double Average3 ( int & rVec //разыменованный константный адрес - значение**

**, int sizeVec) //длина массива**

{

int Sum = 0;

int i;

cout << "/Aver3 : sizeof(rVec) = " << sizeof(rVec) << endl;

for (i = 0; i < sizeVec; i++)

{

**// rVec - значение 0-го элемента массива**

**// &rVec - адрес 0-го элемента массива**

**// &rVec + i - смещение от адреса 0-го элемента массива**

**//\*(&rVec + i) - разыменование по смещению от адреса 0-го элемента**

**//\*(&rVec + i) - ЗНАЧЕНИЕ**

Sum = Sum + \*(&rVec + i);

cout << "/Aver3 : Sum = " << Sum

<< "\t \*(&rVec+" << i << ") = " << \*(&rVec + i)

<< "\t (&rVec+" << i << ") = " << (&rVec + i) << endl;

}//for i

return 1.0 \* Sum / sizeVec; //для преобразования к double

}//Average3()

/\*\*\*\*\*\*\*\* End of Ex04\_05\_ПЕРЕДАЧА МАССИВА.CPP file \*\*\*\*\*\*\*\*/

Теперь при работе с массивом у нас есть два подхода – работать как обычно с массивом через индекс и как с указателем. Указатели являются более общим и широким средством (хотя по-началу не совсем привычным).

1. МНОГОМЕРНЫЕ МАССИВЫ

Специального типа многомерных массивов в С++ не существует. Многомерный массив размерности N можно представить как одномерный массив из массивов размерности N-1. Таким образом, трехмерный массив – это одномерный массив, элементами которого являются матрицы. Многомерные массивы инициализируются в порядке наискорейшего изменения самого правого индекса (как разряды в числах)

**Пример 4.6** Объявление и инициализация многомерных массивов

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*---------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\Ex04\_06 \*

\* File Name : Ex04\_06\_МНОГОМЕРНЫЕ\_МАССИВЫ.CPP \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modifyed By : \*

\* Created : 16/04/04 \*

\* Last Revision : 12/03/19 \*

\* Comment(s) : ПРИМЕР МНОГОМЕРНЫХ МАССИВОВ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include<iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

//setlocale(LC\_ALL, "C"); //отключение русского языка

system("color F0"); //экран белый, буквы черные

system("cls"); //очистка экрана

cout << "\tEx04\_06\_МНОГОМЕРНЫЕ\_МАССИВЫ.CPP\n";

//примеры инициализации

int X[3][2][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24 };

// или

int Y[3][2][4] = { { { 1, 2, 3, 4 } //три двумерных массива

,{ 5, 6, 7, 8 } } //1

,{ { 9, 10, 11, 12 }

,{ 13, 14, 15, 16 } } //2

,{ { 17, 18, 19, 20 }

,{ 21, 22, 23, 24 } } //3

}; //Y[]

int k; //число плоскостей (матриц)

int i; //число строк

int j; //число столбцов

//массивы x и y идентичны

//1

cout << "/1\n";

//массив x

cout << "\n\t Массив X \n";

for (k = 0; k < 3; k++) //по номеру плоскости

{

for (i = 0; i < 2; ++i) //по номеру строки

{

for (j = 0; j < 4; ++j) //по номеру столбца

cout << X[k][i][j] << " "; //по горизонтали

cout << endl;

} //for i

cout << "\n";

} //for k

cout << "\n";

//массив y

cout << "\n\t Массив Y \n";

for (k = 0; k < 3; ++k) //по номеру плоскости

{

for (i = 0; i < 2; ++i) //по номеру строки

{

for (j = 0; j < 4; ++j) //по номеру столбца

{

cout << Y[k][i][j] << " "; //по горизонтали

}//for j

cout << "\n";

} //for i

cout << "\n";

} //for k

cout << "\n";

//2

//**для многомерных массивов можно опускать величину первой размерности**

cout << "/2\n";

int **Z[][3]** = { { 71, 72, 73 } //коды ASCII

,{ 74, 75, 76 } //560 % 256 = 48 код "0"

,{ 77, 78, 79 }

,{ 80, 81, 82 } };

//определяем количество строк

for (i = 0; i < sizeof(Z) / (3\*sizeof(int)); i++) //первая координата

{

for (j = 0; j < 3; j++) //цикл по второй координате

{

cout << " " << Z[i][j] << " " << (char)Z[i][j] << " ";

}//for j

cout << endl;

} //for i

cout << endl;

system("pause");

return 0;

} //end main()

/\*\*\*\*\*\*\* End of Ex04\_06\_МНОГОМЕРНЫЕ\_МАССИВЫ.CPP file \*\*\*\*\*\*/

Чтобы создать функцию, принимающую в качестве аргумента двумерный массив, надо помнить, что имя массива обрабатывается как адрес, то есть является указателем. Надо только правильно объявить указатель.

Пусть функции sum() передается массив matr :

int matr [3][4] = { {1,2,3,4},{9,8,7,6},{2,4,6,8}};

int total = sum (matr, 3);

Как должен выглядеть прототип функции sum()? Почему ей передается только число строк (3), а не число строк (3) и столбцов (4)?

Здесь matr представляет собой имя массива, содержащего 3 элемента, каждый из которых является массивом на 4 элемента. matr указывает на первый из четырех значений первой строки типа int. Прототип выглядит следующим образом:

int sum (int (\*arr2)[4], int size);

здесь int (\*arr2)[4] – указатель на массив из 4 элементов, а

int \*arr2[4] - массив из четырех указателей на данные типа int, а параметр функции не может быть массивом.

int \*arr2[4]; //массив из 4 указателей на элемент типа int

int (\*arr2)[4]; //указатель на массив из 4 int элементов

Существует еще один формат записи прототипа

**int sum (int arr2[][4], int size);**

int sum (int (\*arr2)[4] , int size);

(аналогично int years[ ] и int \*years)

Оба варианта означают, что arr2 представляет собой **указатель**, а не массив. Кроме того, обозначение типа указателя сообщает, что он ссылается на массив из четырех значений типа int. Таким образом, тип указателя уже определяет количество столбцов и отдельным параметром в функцию не передается. Функция sum() может обрабатывать только массивы с четырьмя столбцами, а количество строк может быть любым (!) и задается переменной size (в функции не надо жестко прописывать размер массива).

int sum (int arr2[][4], int size)

{

int total = 0;

for (int i = 0; i < size; i++)

for (int j = 0; j < 4; j++)

total +=arr2[i][j];

return total;

}//sum()

**Пример 4.7** Изменение вектора и матрицы в функции

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* П Я В У С + + \*

\*----------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : Test\_01 \*

\* File Name : Ex04\_13\_ИЗМ\_МАСС\_ФУНКЦИЕЙ.CPP \*

\* Programmer(s) : Чечиков Ю.Б. \*

\* Modifyed By : \*

\* Created : 09/04/14 \*

\* Last Revision : 12/03/19 \*

\* Comment(s) : СТАТИЧЕСКИЕ МАССИВЫ \*

\* ПЕРЕДАЧА ВЕКТОРА В ФУНКЦИЮ. \*

\* **ИЗМЕНЕНИЕ ВЕКТОРА В ФУНКЦИИ**. \*

\* ПЕРЕДАЧА МАТРИЦЫ В ФУНКЦИЮ. \*

\* **ИЗМЕНЕНИЕ МАТРИЦЫ В ФУНКЦИИ**. \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

#include <conio.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Г Л О Б А Л Ь Н Ы Е К О Н С Т А Н Т Ы \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const int N\_MAX = 14; //максимальное число строк в матрице

const int M\_MAX = 15; //максимальное число столбцов в строке (векторе)

const int K\_MAX = 13; //максимальное число матриц в кубе

//const char fNAME[]="SRC1.dat"; //файл исходных данных

//const char fNAME\_Out[]="RES1.dat"; //файл результатов

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//печать вектора

void PrintVec( int \*xVec //указатель на вектор

, int xM); //длина вектора

//изменение вектора

void ChangeVec( int \*xVec //указатель на вектор

, int xM); //длина вектора

//печать матирцы

void PrintMatr( int xMatr[][M\_MAX] //указатель на матрицу

, int xN //число строк

, int xM); //число столбцов

//изменение матрицы

void ChangeMatr( int xMatr[][M\_MAX] //указатель на матрицу

, int xN //число строк

, int xM); //число столбцов

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* О С Н О В Н А Я П Р О Г Р А М М А \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

//якобы считали из файла

int xN = 4; //реальное число строк

int xM = 5; //реальное число столбцов

setlocale(LC\_ALL, "RUSSIAN");

// **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* V E C T O R \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

//якобы считали из файла вектор

int AVec[M\_MAX] = { 1, 2, 3, 4, 5 }; //ШАБЛОН ЧАСТИЧНО ЗАПОЛНЕН

cout << "\t\t В Е К Т О Р \n";

cout << "\n ИСХОДНЫЙ AVec\n";

PrintVec(AVec, xM); //печать вектора

ChangeVec(AVec, xM); //изменение вектора

cout << "\n НОВЫЙ AVec\n";

PrintVec(AVec, xM);//печать вектора

cout << "/main:\t\t"; //печать вектора из основной программы

for (int i = 0; i < xM; i++)

cout << AVec[i] << "\t";

cout << endl;

// **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* M A T R I X \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

//якобы считали из файла матрицу

// int AMatr[4][5] = //матрица НЕЛЬЗЯ !!!

int AMatr[N\_MAX][M\_MAX] = //матрица ШАБЛОН ЧАСТИЧНО ЗАПОЛНЕН

{ { 1, 2, 3, 4, 5 }

,{ 6, 7, 8, 9, 10 }

,{ 11, 12, 13, 14, 15 }

,{ 16, 17, 18, 19, 20 }

};

cout << "\n\n\t\t М А Т Р И Ц А \n";

cout << "\tИСХОДНАЯ AMatr\n";

PrintMatr(AMatr, xN, xM); //печать матрицы

ChangeMatr(AMatr, xN, xM); //изменение матрицы

cout << "\n\tНОВАЯ AMatr\n";

PrintMatr(AMatr, xN, xM); //печать матрицы

//печать матрицы из основной программы

cout << "\n /main:\t\t\n ";

for (int i = 0; i<xN; i++) //по числу строк

{

for (int j = 0; j < xM; j++) //по числу столбцов

{

cout << AMatr[i][j] << " "

<< **\*(\*(AMatr + i) + j)** << "\t";

}//for j

cout << "\n";

}//for i

//\_getch(); //Press any key ...

system("PAUSE");

return 0;

}//main()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*-----------------------------------------------------\*/

/\* печать вектора \*/

/\*----------------\*/

void PrintVec(int \*xVec, int xM)

{

cout << "/PrintVec:\t";

for (int i = 0; i < xM; i++)

**cout << xVec[i] << " " << \*(xVec + i)** << "\t";

cout << endl;

} // end PrintVec()

/\*-----------------------------------------------------\*/

/\* изменение вектора \*/

/\*-------------------\*/

void ChangeVec(int \*xVec, int xM)

{

cout << "\n From ChangeVec()\n\t\t";

for (int i = 0; i < xM; i++)

{

xVec[i] = xVec[i] \* 2; //удвоили значение

cout << xVec[i] << "\t";

} // end for i

cout << endl;

} //end ChangeVec()

/\*-----------------------------------------------------\*/

/\* печать матрицы \*/

/\*----------------\*/

void PrintMatr( int xMatr[][M\_MAX] //указатель на матрицу

, int xN //число строк

, int xM //число столбцов

)

{

int i, j;

cout << "\t/FROM PrintMatr: \n";

//печатаем матрицу

for (i = 0; i < xN; i++) //по числу строк

{

for (j = 0; j < xM; j++) //по числу столбцов

{

cout << xMatr[i][j] << " "

<< **\*(\*(xMatr + i) + j)** << "\t";

} //for j

cout << "\n";

}//for i

} //end PrintMatr()

/\*-----------------------------------------------------\*/

/\* изменение матрицы \*/

/\*-------------------\*/

void ChangeMatr( int xMatr[][M\_MAX] //указатель на матрицу

, int xN //число строк

, int xM //число столбцов

)

{

int i, j;

cout << "\n\n\t/FROM ChangeMatr: \n";

//меняем и печатаем матрицу

for (i = 0; i < xN; i++) //по числу строк

{

for (j = 0; j < xM; j++) //по числу столбцов

{

**xMatr[i][j] \*= 2;** //удвоили значение

cout << xMatr[i][j] << "\t"; //через индекс

}//for j

cout << "\n";

}//for i

} //end ChangeMatr()

/\*\*\*\*\*\*\*\*\* End Of Ex04\_13\_ИЗМ\_МАСС\_ФУНКЦИЕЙ.CPP File \*\*\*\*\*\*\*/

**Пример 4.8** Умножение векторов и матриц

//#include "stdafx.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* КАФЕДРА № 302 1 КУРС ПЯВУ \*

\*-------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : Ex04\_Matrix\_01 \*

\* File Name : Ex04\_УМНОЖЕНИЕ МАТРИЦ через ФУНКЦИИ \*

\* Language : C/C++ \*

\* Programmer(s) : Чечиков Ю.Б. & Татарникова Е.М. \*

\* Modifyed By : \*

\* Created : 10/03/15 \*

\* Last Revision : 01/04/19 \*

\* Comment(s) : ПРИМЕРЫ РАБОТЫ С ДВУМЕРНЫМИ МАССИВАМИ \*

\* РЕАЛЬНЫЙ РАЗМЕР МЕНЬШЕ ЗАКАЗАННОГО \*

\* УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР \*

\* УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР \*

\* УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ \*

\* ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ \*

\* ЧЕРЕЗ ФУНКЦИИ \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Г Л О Б А Л Ь Н Ы Е К О Н С Т А Н Т Ы \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//размеры ШАБЛОНА

const int N\_MAX = 15; //максимальное число строк в матрице

const int M\_MAX = 15; //максимальное число столбцов в строке (векторе)

const int K\_MAX = 15; //максимальное число столбцов в строке

//

////const char fNAME[]="SRC1.dat"; //файл исходных данных

////const char fNAME\_Out[]="RES1.dat"; //файл результатов

//

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**//печать вектора**

void PrintVec( int \*xVec //указатель на вектор

, int xM); //длина вектора

**//умножение вектора на вектор**

int Mult\_VecVec( int \*xVec //указатель на вектор

, int \*yVec //указатель на вектор

, int xM); //длина вектора

**//печать матрицы**

void PrintMatr( int xMatr[][M\_MAX] //указатель на матрицу

, int xN //число строк

, int xM); //число столбцов

**//умножение матрицы на вектор**

void Mult\_MatrVec(int xMatr[][M\_MAX] //указатель на матрицу

, int \*xVec //указатель на вектор

, int xN //число строк

, int xM //число столбцов

, int \*xResVec); //указатель на вектор

////умножение матрицы на вектор - указатель на результат-вектор

//int \* Mult\_MatrVecP(int xMatr[][M\_MAX] //указатель на матрицу

// , int \*xVec //указатель на вектор

// , int xN //число строк

// , int xM); //число столбцов

////ДЛЯ ДИНАМИЧЕСКИХ МАССИВОВ !!!

////умножение матрицы на вектор - указатель на вектор

//void Mult\_MatrVec\_P2( int \*\*xMatr //указатель на матрицу

// , int \*xVec //указатель на вектор

// , int xN //число строк

// , int xM //число столбцов

// , int \*xResVec); //указатель на вектор

**//умножение матрицы на матрицу**

void Mult\_MatrMatr(int xMatr[][M\_MAX]//указатель на матрицу [N\_MAX\*M\_MAX]

, int yMatr[][K\_MAX] //указатель на матрицу [M\_MAX\*K\_MAX]

, int xN //число строк

, int xM //число столбцов/строк

, int xK //число столбцов

, int xResMult\_MM[][K\_MAX] //ук. на матрицу [N\_MAX\*K\_MAX]

);

**// транспонирование матрицы самой в себя**

void TranspMatr( int xMatr[][M\_MAX] //указатель на матрицу

, int xNM); //число строк и столбцов

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* О С Н О В Н А Я П Р О Г Р А М М А \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

int main()

{

//стандартная молитва

setlocale(LC\_ALL, "RUSSIAN"); //подключение русского языка

system("color F0"); //экран белый, буквы черные

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{ //block УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР

//локальные константы

const int M = 5; //число столбцов (длина вектора)

//объявление переменных

int AVec[M] = { 1, 2, 3, 4, 5 }; //вектор A

int BVec[M] = { 1, 5, 10, 20, 10 }; //вектор B

int ResMult\_VV = 0; //результат умножения вектора на вектор

cout << "\tУМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР\n";

//печать исходных данных

cout << "\tВЕКТОР А\n";

PrintVec(AVec, M); //печать вектора

cout << "\tВЕКТОР B\n";

PrintVec(BVec, M); //печать вектора

//умножение вектора на вектор

ResMult\_VV = Mult\_VecVec(AVec, BVec, M);

//печать результата

cout << "\n\tРЕЗУЛЬТАТ УМНОЖЕНИЯ ВЕКТОРА НА ВЕКТОР = "

<< ResMult\_VV << "\n\n";

system("pause");

/\*

Работа программы.

УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР

ВЕКТОР А

1 2 3 4 5

ВЕКТОР B

1 5 10 20 10

РЕЗУЛЬТАТ УМНОЖЕНИЯ ВЕКТОРА НА ВЕКТОР = 171

\*/

}//end block УМНОЖЕНИЕ ВЕКТОРА НА ВЕКТОР

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{

//Умножение матрицы размером N\*M на вектор размером M.

//[N \* M] - реальный размер матрицы AMatr

//якобы взято из файла

const int N = 2; //якобы реальное число строк - взято из файла

const int M = 3; //якобы реальное число столбцов (длина вектора)

//объявление переменных

//[N \* M] - реальный размер матрицы AMatr,

//а память выделена [N\_MAX \* M\_MAX]

int AMatr[N\_MAX][M\_MAX] = { { 1, 2, 3 } //1 строка

, { 3, 2, 1 } //2 строка

}; //AMatr

int AVec[M\_MAX] = { 4, 5, 6 }; //вектор A

int ResVec[N]; //результат умножения матрицы на вектор

cout << "\n\t\tУМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР\n";

/\*

//ввод исходных данных

cout << "\tВВОД МАТРИЦЫ AMatr\n";

for (i = 0; i < N; i++)

{

cout << "Введите " << i << "- ю строку матрицы из " << M << " элементов\n";

for (j = 0; j < M; j++)

{

//cout <<"AMatr[" << i << "][" << j << "]= ";

cin >> AMatr[i][j];

//cout << "\t" << AMatr[i][j]<< "\t";

}; // for j

cout << endl;

}; //for i

cout << "\tВВОД ВЕКТОРА-столбца AVec\n";

for (j = 0; j < M; j++)

{

cout << "AVec[" << j << "]= ";

cin >> AVec[j];

//cout << "\t" << AVec[j]<< "\t";

}; //for j

cout << endl;

\*/

//Печать исходных данных

cout << "\t МАТРИЦА AMatr\n";

//[N \* M] - реальный размер матрицы AMatr

// [M] - реальный размер вектора AVec

//[N] - реальный размер вектора ResVec

PrintMatr(AMatr, N, M);

cout << "\tВЕКТОР AVec\n";

PrintVec(AVec, M);

// УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР

cout << "\tРЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА ВЕКТОР\n";

Mult\_MatrVec(AMatr, AVec, N, M, ResVec);

PrintVec(ResVec, N);

//НЕ РАБОТАЕТ - разные области памяти pResVec и xResVec в стеке

//int \* pResVec = Mult\_MatrVecP(AMatr, AVec, N, M);

//cout << "\tРЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА ВЕКТОР Point\n";

//PrintVec(pResVec, N);

//НЕ РАБОТАЕТ - ДВОЙНОЙ УКАЗАТЕЛЬ \*\*xMatr

//и указатель на вектор AMatr[][M\_MAX] !!! - разные типы !

//cout << "\t/2 РЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА ВЕКТОР Point\n";

//Mult\_MatrVec\_P2(AMatr, AVec, N, M, ResVec);

//PrintVec(ResVec, N);

system("pause");

/\*

Работа программы.

УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР

МАТРИЦА AMatr

1 2 3

3 2 1

ВЕКТОР AVec

4 5 6

РЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА ВЕКТОР

32 28

\*/

} // end block УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{ //block УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

//локальные константы

const int N = 4; //число строк

const int M = 3; //число столбцов (длина вектора)

const int K = 2;

//объявление переменных

int AMatr[N\_MAX][M\_MAX] = { { 1, 2, 3 }, { 2, 3, 4 }

, { 4, 3, 2 }, { 3, 2, 1 } };

int BMatr[M\_MAX][K\_MAX] = { { 1, 0 }, { 0, 1 }, { 1, 2 } };

int ResMult\_MM[N\_MAX][K\_MAX];//результат умножения MM

cout << "\n\t\tУМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ\n";

//[N \* M] - реальный размер матрицы AMatr

//[M \* K] - реальный размер матрицы BMatr

//[N \* K] - реальный размер матрицы ResMult\_MM

/\*

//ввод исходных данных

cout << "\tВВОД МАТРИЦЫ AMatr\n";

for (i = 0; i < N; i++) //по числу строк

{

cout << "Введите " << i << "-ю строку матрицы из " << M << " элементов\n";

for (j = 0; j < M; j++) //по длине строки

{

cin >> AMatr[i][j];

}; // for j

// cout << endl;

}; //for i

cout << endl;

cout << "\tВВОД МАТРИЦЫ BMatr\n";

for (j = 0; j < M; j++) //по числу строк

{

cout << "Введите " << j << "-ю строку матрицы из " << K << " элементов\n";

for (k = 0; k < K; k++) //по длине строки

{

cin >> BMatr[j][k];

}; // for j

// cout << endl;

}; //for i

cout << endl;

\*/

//Печать исходных данных

cout << "\t МАТРИЦА AMatr\n";

PrintMatr(AMatr, N, M); //печать матрицы

cout << "\t МАТРИЦА BMatr\n";

PrintMatr(BMatr, M, K); //печать матрицы

cout << "\tРЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА МАТРИЦУ \n";

Mult\_MatrMatr(AMatr, BMatr, N, M, K, ResMult\_MM);

PrintMatr(ResMult\_MM, N, K); //печать матрицы

system("pause");

/\*

Работа программы.

УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

МАТРИЦА AMatr

1 2 3

2 3 4

4 3 2

3 2 1

МАТРИЦА BMatr

1 0

0 1

1 2

РЕЗУЛЬТАТ УМНОЖЕНИЯ МАТРИЦЫ НА МАТРИЦУ

4 8

6 11

6 7

4 4

Для продолжения нажмите любую клавишу . . .

1 2 3 1 0 0 1 2 3

3 2 1 умножить 0 1 0 равно 3 2 1

0 0 1

Результат

1 2 3

3 2 1

\*/

}//end block УМНОЖЕНИЕ МАТРИЦЫ НА МАТРИЦУ

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ \*/**

**/\* (без использования дополнительного массива) \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

{ //block ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ

//локальные константы

const int NM = 5; //число строк и столбцов (квадратная матр)

//объявление переменных

//лучше, чтобы N\_MAX = M\_MAX. {..} - обязательно !!

int AMatr[N\_MAX][M\_MAX] = { { 11, 12, 13, 14, 15 }

, { 21, 22, 23, 24, 25 }

, { 31, 32, 33, 34, 35 }

, { 41, 42, 43, 44, 45 }

, { 51, 52, 53, 54, 55 } };

cout << "\n\t\tТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ\n";

/\*

//ввод исходных данных

cout << "\tВВОД МАТРИЦЫ AMatr\n";

for (i = 0; i < N; i++)

{

cout << "\tВведите " << i << "- ю строку матрицы из "

<< N << " элементов\n";

for (j = 0; j < N; j++)

{

cin >> AMatr[i][j];

}; // for j

cout << endl;

}; //for i

\*/

//Печать исходных данных

cout << "\n\t МАТРИЦА AMatr\n";

PrintMatr(AMatr, NM, NM); //печать матрицы

//ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ

TranspMatr(AMatr, NM);

cout << "\n\tТРАНСПОНИРОВАННАЯ МАТРИЦА AMatr\n";

PrintMatr(AMatr, NM, NM); //печать матрицы

system("pause");

/\*

Работа программы.

ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ

МАТРИЦА AMatr

11 12 13 14 15

21 22 23 24 25

31 32 33 34 35

41 42 43 44 45

51 52 53 54 55

ТРАНСПОНИРОВАННАЯ МАТРИЦА AMatr

11 21 31 41 51

12 22 32 42 52

13 23 33 43 53

14 24 34 44 54

15 25 35 45 55

Для продолжения нажмите любую клавишу . . .

\*/

} // end block ТРАНСПОНИРОВАТЬ МАТРИЦУ САМУ В СЕБЯ

system("pause");

return 0;

} //end main()

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\*-----------------------------------------------------\*/**

**/\* печать вектора \*/**

**/\*----------------\*/**

void PrintVec(int \*xVec //указатель на вектор

, int xM) //длина вектора

{

for (int j = 0; j < xM; j++)

{

cout << " " << xVec[j];

} //end for j

cout << "\n";

}//PrintVec()

**/\*-----------------------------------------------------\*/**

**/\* умножение вектора \*/**

**/\* на вектор \*/**

**/\*-------------------\*/**

int Mult\_VecVec(int \*xVec //указатель на вектор

, int \*yVec //указатель на вектор

, int xM) //длина вектора

{

int ResMult\_VV = 0; //результат умножения двух векторов - число

int j;

//умножение вектора на вектор

for (j = 0; j < xM; j++)

{

ResMult\_VV = ResMult\_VV + xVec[j] \* yVec[j];

} //end for i

return ResMult\_VV;

}//Mult\_VecVec ()

**/\*-----------------------------------------------------\*/**

**/\* печать матрицы \*/**

**/\*----------------\*/**

void PrintMatr(int xMatr[][M\_MAX] //указатель на матрицу

, int xN //число строк

, int xM) //число столбцов

{

//cout << " /PrintMatr: xN = "<< xN << " xM = " << xM << endl;

for (int i = 0; i < xN; i++)

{

for (int j = 0; j < xM; j++)

{

cout << " " << xMatr[i][j]; //печать строки

}; // for j

cout << endl;

}; //for i

// cout << endl;

} //PrintMatr()

**/\*-----------------------------------------------------\*/**

**/\* умножение матрицы \*/**

**/\* на вектор \*/**

**/\*-------------------\*/**

void Mult\_MatrVec(int xMatr[][M\_MAX] //указатель на матрицу

, int \*xVec //указатель на вектор

, int xN //число строк

, int xM //число столбцов

, int \*xResVec) //указатель на вектор

{

int i, j;

for (i = 0; i < xN; i++) //выбор вектора

{

**//ЯВНО РАБОТАЕТ**

**//xResVec[i] = 0; //умножение вектора на вектор**

**//for (j = 0; j < xM; j++)**

**//{**

**// xResVec[i] = xResVec[i] + xMatr[i][j] \* xVec[j];**

**//}//for j**

//ИЛИ через ФУНКЦИЮ

**xResVec[i] = Mult\_VecVec(xMatr[i], xVec, xM); //РАБОТАЕТ**

// cout << "\t\t" << xResVec[i] << endl;

}//for i

// cout << endl;

}//Mult\_MatrVec()

**/\*-----------------------------------------------------\*/**

**/\* умножение матрицы \*/**

**/\* на матрицу \*/**

**/\*-------------------\*/**

void Mult\_MatrMatr(int xMatr[][M\_MAX]//указатель на матрицу [N\_MAX\*M\_MAX]

, int yMatr[][K\_MAX]//указатель на матрицу [M\_MAX\*K\_MAX]

, int xN //число строк

, int xM //число столбцов/строк

, int xK //число столбцов

, int xResMult\_MM[][K\_MAX])//ук. на матрицу [N\_MAX\*K\_MAX]

{

int i, j, k;

for (k = 0; k < xK; k++) //цикл по выбору столбца из матрицы B

{

**//умножение матрицы на вектор-столбец РАЗОБРАТЬСЯ !!!**

**//Mult\_MatrVec(xMatr, yMatr[k], xN, xM, xResMult\_MM[i]);**

//умножение матрицы на вектор-столбец

for (i = 0; i < xN; i++) //строка из матрицы A

{

**//xResMult\_VV РАЗОБРАТЬСЯ !!!!**

//РАБОТАЕТ

xResMult\_MM[i][k] = 0; //умножаем два вектора

for (j = 0; j < xM; j++) //по длине векторов

{

xResMult\_MM[i][k] = xResMult\_MM[i][k]

+ xMatr[i][j] \* yMatr[j][k];

}//for j

} // for i

} //for k

}//Mult\_MatrMatr()

/\*-----------------------------------------------------\*/

/\* умножение матрицы \*/

/\* на вектор 2 \*/

/\*-------------------\*/

////умножение матрицы на матрицу - указатель на вектор

//int \* Mult\_MatrVecP(int xMatr[][M\_MAX] //указатель на матрицу

// , int \*xVec //указатель на вектор

// , int xN //число строк

// , int xM) //число столбцов

//{

// int xResVec[M\_MAX];

// for (int i = 0; i < xN; i++)

// {

// xResVec[i] = 0;

// for (int j = 0; j < xM; j++)

// {

// xResVec[i] = xResVec[i] + xMatr[i][j] \* xVec[j];

// }//for j

//// cout << "\t\t" << xResVec[i] << endl;

// }//for i

// cout << endl;

// return xResVec; //НЕ СРАБОТАЕТ !!! - локальная переменная

//}//Mult\_MatrVecP()

// //умножение матрицы на матрицу - указатель на вектор

//void Mult\_MatrVec\_P2(int \*\*xMatr //указатель на матрицу

// , int \*xVec //указатель на вектор

// , int xN //число строк

// , int xM //число столбцов

// , int \*xResVec) //указатель на вектор

//{

// //int xResVec[M\_MAX];

// for (int i = 0; i < xN; i++) //выбор вектора xMatr[i]

// {

// xResVec[i] = 0;

// for (int j = 0; j < xM; j++)

// { //умножаем вектор xMatr[i] на вектор xVec

// xResVec[i] = xResVec[i] + xMatr[i][j] \* xVec[j];

// }//for j

// // cout << "\t\t" << xResVec[i] << endl;

// }//for i

// cout << endl;

//

//}//Mult\_MatrVec\_P2()

**/\*-----------------------------------------------------\*/**

**/\* транспонирование матрицы \*/**

**/\* самой в себя \*/**

**/\*--------------------------\*/**

void TranspMatr(int xMatr[][M\_MAX]

, int xNM)

{

int Temp;

for (int i = 0; i < xNM; i++)

{

//элементы на главной диагонали не трогаем

//for (**j = 0**; j < xNM; j++) //с ошибкой!!!!

for (int **j = i + 1**; j < xNM; j++) //без ошибки

{

Temp = xMatr[i][j];

xMatr[i][j] = xMatr[j][i];

xMatr[j][i] = Temp;

}; // for j

}; //for i

}//TranspMatr()

/\*\*\*\* End Of Ex04\_УМНОЖЕНИЕ МАТРИЦ через ФУНКЦИИ.CPP FILE \*\*\*\*/

1. ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ

**ОПЕРАТОРЫ new и delete.**

Переменная является *именованной* областью памяти, которая резервируется во время компиляции (статическая область памяти; имя переменной – это значение, которое хранится где-то в памяти; & - адрес), а указатели всего лишь служат псевдонимами (второе имя) для областей памяти, к которым в любом случае можно обратиться напрямую по имени (указатель – переменная, хранящая адрес значения вместо самого значения). Но истинная ценность указателей проявляется тогда, когда для хранения данных выделяется **неименованная** область памяти *во время* выполнения программы (динамическая область памяти). В этом случае указатели становятся единственным способом доступа к этим областям памяти. Для динамического выделения памяти используется оператор new .

Оператору new мы указываем для какого типа данных (сколько байт, фактически) выделяется память. Оператор new ищет блок памяти нужного размера и возвращает адрес этого блока. Выделение памяти происходит **при выполнении** программы. Присваивая этот адрес указателю, мы можем работать с выделенным блоком. Общая форма записи для выделения памяти под основной тип или тип *структура* имеет вид:

ИмяТипа \*ИмяУказателя = new ИмяТипа;

Например:

int \*pn = new int; или int \*pn; pn =new int;

Для *освобождения* выделенной памяти (возврат ее в общую область) используется оператор delete . Освобожденная память может быть повторно использована в других местах другими частями программы. В тексте программы за оператором delete должен следовать указатель на блок памяти, ранее зарезервированный при помощи оператора new . Например:

delete pn;

В данном случае очищается область памяти, на которую указывает pn , но сам pn **не удаляется**. Его можно использовать повторно для указания на другую выделенную область памяти **того же** типа. Операторы new и delete всегда нужно использовать в паре, иначе может произойти утечка памяти, что может привести к останову программы при поиске дополнительной памяти.

Нельзя повторно очищать блок памяти, который уже был освобожден. Результат такого действия неопределен. Нельзя использовать оператор delete для освобождения памяти, созданной при объявлении переменной (статическая память):

delete pn; //допустимо

delete pn; //уже недопустимо

int Temp = 5; //допустимо

int \*pT = &Temp; //допустимо

delete pT //недопустимо, память не выделена

//оператором new

**Пример 4.9** Динамическое выделение памяти для переменных

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*---------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : Ex04\_07 \*

\* File Name : Ex04\_07 ДИНАМИЧЕСКАЯ ПАМЯТЬ ДЛЯ ПЕРЕМЕННЫХ.cpp\*

\* Programmer(s) : Чечиков Ю.Б. & Безродных И.П. \*

\* Modifyed By : \*

\* Created : 20/04/04 \*

\* Last Revision : 21/03/19 \*

\* Comment(s) : ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ И \*

\* ОСВОБОЖДЕНИЕ ПАМЯТИ ДЛЯ ПЕРЕМЕННЫХ \*

\*---------------------------------------------------------------\*

\* Результат работы программы \*

\* ОПЕРАТОР NEW ДЛЯ ПЕРЕМЕННЫХ !! \*

\* \*

\* АДРЕС pn = 0000024F6C079C90 \*

\* int VALUE \*pn = 1235 \*

\* double VALUE \*pd = 2.71828 \*

\* АДРЕС pd = 0000024F6C078AB0 \*

\* \*

\* size of \*pn = 4 size of pn = 8 \*

\* size of \*pd = 8 size of pd = 8 \*

\* \*

\* ПОСЛЕ delete pn \*

\* АДРЕС pn = 0000000000008123 \*

\* \*

\* ОБЪЯВИМ double \* pd2 = pd \*

\* АДРЕС pd2 = 0000024F6C078AB0 \*

\* double VALUE \*pd2 = 2.71828 \*

\* \*

\* ПОСЛЕ delete pd2 \*

\* АДРЕС pd = 0000024F6C078AB0 \*

\* double VALUE \*pd = -1.45682e+144 \*

\* АДРЕС pd2 = 0000000000008123 \*

\* \*

\* ПОВТОРНОЕ ИСПОЛЬЗОВАНИЕ УКАЗАТЕЛЯ pn \*

\* АДРЕС pn = 0000024F6C079350 \*

\* int VALUE \*pn = 55555 \*

\* size of pn = 8 size of \*pn = 4 \*

\* \*

\* Для продолжения нажмите любую клавишу . . . \*

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "RUSSIAN");

cout << "\t ОПЕРАТОР NEW ДЛЯ ПЕРЕМЕННЫХ !! \n\n";

//выделение области памяти для ЗНАЧЕНИЯ типа int

**int \* pn = new int;**

\*pn = 1235; //собственно значение

cout << " АДРЕС pn = " << pn << endl;

cout << " int VALUE \*pn = " << \*pn << endl;

**double \* pd = new double; //выделение области памяти для double**

\*pd = 2.718281828; //собственно значение

cout << " double VALUE \*pd = " << \*pd<< endl;

cout << " АДРЕС pd = " << pd << "\n\n";

cout << " size of \*pn = " << sizeof \*pn

<< "\tsize of pn = " << sizeof pn << endl;

cout << " size of \*pd = " << sizeof \*pd

<< "\tsize of pd = " << sizeof pd << "\n\n";

**//system("PAUSE");**

//освобождаем память, НО САМ УКАЗАТЕЛЬ НЕ УДАЛЯЕТСЯ

**delete pn; //очищается область памяти**

cout << " ПОСЛЕ delete pn \n";

cout << " АДРЕС pn = " << pn << "\n\n";

//cout << "\t int VALUE \*pn = " << \*pn << endl; //ОШИБКА !!

**//system("PAUSE");**

double \* pd2 = pd;

cout << " ОБЪЯВИМ double \* pd2 = pd\n";

cout << " АДРЕС pd2 = " << pd2 << "\n";

cout << " double VALUE \*pd2 = " << \*pd2<< "\n\n";

//освобождаем АДРЕС, на который укзывал указатель, а НЕ УКАЗАТЕЛЬ !

**delete pd2;**

cout << " ПОСЛЕ delete pd2 \n";

cout << " АДРЕС pd = " << pd << "\n";

cout << " double VALUE \*pd = " << \*pd << "\n";

cout << " АДРЕС pd2 = " << pd2 << "\n\n";

//cout << " double VALUE \*pd2 = " << \*pd2 << "\n"; //ОШИБКА !!

// **int \*pint = new float; //менять тип pn НЕЛЬЗЯ !!!**

// float \*pn = new float; //менять тип pn НЕЛЬЗЯ !!!

//повторное использование указателя на динамическую память

cout << "\tПОВТОРНОЕ ИСПОЛЬЗОВАНИЕ УКАЗАТЕЛЯ pn\n";

**pn = new int;**

\*pn = 55555; //собственно значение

cout << " АДРЕС pn = " << pn << endl;

cout << " int VALUE \*pn = " << \*pn << endl;

cout << " size of pn = " << sizeof pn

<< "\tsize of \*pn = " << sizeof \*pn << "\n\n";

//освобождаем память, НО САМ УКАЗАТЕЛЬ НЕ УДАЛЯЕТСЯ

delete pn; //очищается область памяти

**//delete pd; //ERROR повторное освобождение pd = pd2**

system("PAUSE");

return 0;

}//main()

/\*\* End of Ex04\_07 ДИНАМИЧЕСКАЯ ПАМЯТЬ ДЛЯ ПЕРЕМЕННЫХ.CPP file \*\*/

1. ОПЕРАТОР new ДЛЯ СОЗДАНИЯ ДИНАМИЧЕСКИХ МАССИВОВ

Предположим, что создается программа, для которой массив может либо потребоваться, либо нет в зависимости от информации, полученной в ходе работы программы. Длина массива будет определяться после запуска программы.

Если массив создается путем объявления, то память под него отводится на этапе компиляции, независимо от того будет использоваться массив в программе или нет. Массив уже встроен в программу и размер его заранее определен. Выделение памяти для массива в процессе компиляции называется *статическим связыванием*. Оператор new позволяет создавать или не создавать массив в зависимости от потребности. Можно даже выбирать размер массива в процессе выполнения программы. Процесс создания массива в ходе выполнения программы называется *динамическим связыванием*, а сам массив – *динамическим*.

При создании динамического массива необходимо указать оператору new тип элементов массива и их количество. Общая форма выделения и присвоения области памяти для массива имеет вид:

ИмяТипа \*ИмяУказателя = new ИмяТипа[ЧислоЭлементов];

В результате выполнения оператора new резервируется блок памяти, достаточный для хранения элементов типа ИмяТипа в количестве ЧислоЭлементов , а указатель ИмяУказателя устанавливается на первый элемент массива.

Например:

int \*pnArr = new int [10]; //блок на 10 значений типа int

Для высвобождения памяти, выделенной под динамический массив с помощью оператора new , используется другая форма оператора delete .

delete [ ] pnArr;

Квадратные скобки указывают, что необходимо высвободить память, отведенную для *целого* массива, а не только *элемента*, на который указывает указатель pnArr (delete pnArr;).

pnArr является указателем на *один* элемент типа int – первый элемент блока из 10 элементов. Так как для динамического массива оператор sizeof не работает, то программист должен сам реализовывать проверку количества элементов массива в программе.

A1

A1

Адрес

Зн 1

Значение

int \*pVec = new int [10]; //вектор

Зн 1

Зн 2

Зн m

**Пример 4.10** Динамическое выделение памяти для массивов

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*--------------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : Ex04\_08 \*

\* File Name : Ex04\_08\_ДИНАМИЧЕСКАЯ ПАМЯТЬ ДЛЯ ВЕКТОРА.CPP \*

\* Programmer(s) : Чечиков Ю.Б. & Безродных И.П. \*

\* Modifyed By : \*

\* Created : 20/04/04 \*

\* Last Revision : 12/03/19 \*

\* Comment(s) : ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ ДЛЯ ВЕКТОРА \*

\*--------------------------------------------------------------------\*

\* Результат работы программы \*

\* \*

\* ОПЕРАТОР NEW[] ДЛЯ ВЕКТОРА \*

\* ВВЕДИТЕ ЧИСЛО ЭЛЕМЕНТОВ N = 5 \*

\* \*

\* ДИНАМИЧЕСКИЙ МАССИВ \*

\* Адрес = 01EDC1E74260 Значение pdVec[0] = 1 \*(pdVec + 0) = 1 \*

\* Адрес = 01EDC1E74268 Значение pdVec[1] = 0.5 \*(pdVec + 1) = 0.5 \*

\* Адрес = 01EDC1E74270 Значение pdVec[2] = 0.33 \*(pdVec + 2) = 0.33 \*

\* Адрес = 01EDC1E74278 Значение pdVec[3] = 0.25 \*(pdVec + 3) = 0.25 \*

\* Адрес = 01EDC1E74280 Значение pdVec[4] = 0.2 \*(pdVec + 4) = 0.2 \*

\* \*

\* ПОСЛЕ delete[] pdVec \*

\* Для продолжения нажмите любую клавишу . . . \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

setlocale(LC\_ALL, "RUSSIAN");

cout << "\tОПЕРАТОР NEW[] ДЛЯ ВЕКТОРА \n";

int n, i;

cout << " ВВЕДИТЕ ЧИСЛО ЭЛЕМЕНТОВ N = ";

cin >> n;

//место хранения значения

double \* pdVec = new double[n]; // pdVec трактуется как имя массива

// и как указатель на начало массива

// sizeof не работает

//заполнение массива

for (i = 0; i < n; i++)

{

pdVec[i] = 1.0 / (i + 1);

}//for i

// печать массива

cout << " ДИНАМИЧЕСКИЙ МАССИВ\n";

for (i = 0; i < n; i++)

{

cout << setprecision(2)

<< " Адрес = " << pdVec + i

<< " Значение pdVec[" << i << "] = " << pdVec[i]

<< "\t\*(pdVec + " << i << ") = " << \*(pdVec + i)

<< "\n";

}//for i

cout << "\n\n";

//удаление динамического массива

delete[] pdVec;

cout << "\t ПОСЛЕ delete[] pdVec \n";

**// НЕЛЬЗЯ !! программа вылетает**

//cout << " pdVec[0] Значение = " << pdVec[0]

// << "\t Адрес = " << &pdVec[0] << "\n";

//cout << " pdVec[1] Значение = " << pdVec[1]

// << "\t Адрес = " << &pdVec[1] << "\n\n";

system("PAUSE");

return 0;

}//main()

/\*\*\*\* End of Ex04\_08\_ДИНАМИЧЕСКАЯ ПАМЯТЬ ДЛЯ ВЕКТОРА.CPP file \*\*\*/

int \*\*ppMatr; //двойной указатель на матрицу n\*m

int \*\*ppMatr; //двойной указатель на матрицу n\*m

ppMatr = new int \* [n];//массив указателей

ppMatr

А1

А2

Аn

int \*\*ppMatr; //двойной указатель на матрицу

ppMatr = new int \* [n];//массив указателей

for (i=0; i<n; ++i)

ppMatr[i] = new int [m];//массив чисел

ppMatr

ppMatr

А1

А2

Аn

Зн 1

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

Зн 2

Зн m

А1

А2

Аk

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

ppMatr

А1

А1

А2

Аk

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

Аn

А2

int \*\*\*pppCube; //тройной указатель на куб k\*n\*m

pppCube = new int \*\* [k];//массив указателей на

//2-мерный массив n\*m

for (l=0; l<k; ++l) //по числу матриц

{ pppCube [l] = new int \* [n];//массив указателей на

// строки

for (i=0;i<n;++i) //по числу строк

pppCube[l][i] = new int [m]; ];//массив чисел

}//for l

А1

А2

Аk

Зн 2

Зн m

Зн 1

Зн 2

Зн m

Зн 1

Зн 2

Зн 1

Зн m

**Пример 4.11** Чтение динамических массивов средствами С++

разных размерностей из файла, передача массивов в функцию, запись массива в файл. средствами С++

Подготовить в текстовом редакторе файлы исходных данных:

**SRC\_VEC.txt**

4

10 11 12 13

**SRC\_MATR.txt**

3 5

11 12 13 14 15

21 22 23 24 25

31 32 33 34 35

//#include "stdafx.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*----------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : Ex04\_15 \*

\* File Name : Ex04\_15\_ДИНАМИЧ\_МАТР\_File\_IO\_С++.cpp \*

\* Programmer(s) : Чечиков Ю.Б. & Безродных И.П. \*

\* Modifyed By : \*

\* Created : 06/06/08 \*

\* Last Revision : 03/04/19 \*

\* Comment(s) : **ДИНАМИЧЕСКИЕ МНОГОМЕРНЫЕ МАССИВЫ** \*

\* УНИВЕРСАЛЬНАЯ РАБОТА С МАССИВОМ ЧЕРЕЗ УКАЗАТЕЛЬ\*

**\* ВВОД/ВЫВОД В ФАЙЛ СРЕДСТВАМИ C++** \*

\* ЧТЕНИЕ РАЗМЕРА ВЕКТОРА И ВЕКТОРА ИЗ ФАЙЛА \*

\* ЧТЕНИЕ РАЗМЕРА МАТРИЦЫ И МАТРИЦЫ ИЗ ФАЙЛА \*

\* ЗАПИСЬ МАТРИЦЫ В ФАЙЛ \*

\* ПЕРЕДАЧА ВЕКТОРА И МАТИРИЦЫ В ПРОЦЕДУРУ \*

\* ВСЕ ФАЙЛЫ ТЕКСТОВЫЕ \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// читаем данные из файла

#include <iostream>

#include <fstream>

using namespace std;

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Г Л О Б А Л Ь Н Ы Е К О Н С Т А Н Т Ы \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

const char fNAME\_VEC[] = "SRC\_VEC.txt"; //файл исходных данных

const char fNAME\_MATR[] = "SRC\_MATR.txt"; //файл исходных данных

const char fNAME\_Out[] = "RES\_MATR.dat"; //файл результатов

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**//вектор**

//ввод вектора через указатель

int InputVec( const char \*fNAME\_VEC

, int \* \*pVec

, int \*xM);

//печать вектора

void PrintVec( int xVec[]

, int xM); //(int \*pVec, int xM);

//освободить память

void ReleaseVec(int \*pVec);

**//матрица**

//ввод матрицы из файла через указатель

int InputMatr( const char \*fNAME\_MATR

, int \* \*\*ppMatr

, int &xN

, int &xM);

//печать матрицы

void PrintMatr( int \*\*ppMatr

, int xN

, int xM);

// Запись матрицы в файл

void WMatrToFile( const char \*xfNAME\_Out\_MATR

, int \*\*ppxMatr

, int xN

, int xM);

//освободить память

void ReleaseMatr( int \*\*ppxMatr

, int xN);

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* О С Н О В Н А Я П Р О Г Р А М М А \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

int main()

{

FILE \*fp = NULL; // объявим указатель на входной файл

FILE \*fp\_out = NULL; // объявим указатель на выходной файл

// int k; //число плоскостей

int n; //число строк

int m; //число столбцов

int TotSum;

int ErrorCode;

//==========================================================

cout << "\n\t VECTOR \n";

TotSum = 0;

int \* pVec = 0; //указатель на вектор

//чтение вектора из файла

ErrorCode = InputVec(fNAME\_VEC, &pVec, &m);

if (ErrorCode)

{

cout << "\nERROR DATA\n" << endl;

cin.get();

return -1;

} //if

PrintVec(pVec, m); //печать вектора из процедуры

ReleaseVec(pVec); //освобождение памяти

//==========================================================

cout << "\n\n\t MATRIX \n";

TotSum = 0;

//указатель на указатель, который указывает на целое число

int \*\*ppMatr = 0; //двойной указатель на массив целых

//чтение матрицы из файла

ErrorCode = InputMatr(fNAME\_MATR, &ppMatr, n, m);

if (ErrorCode)

{

cout << "\nERROR DATA\n" << endl;

cin.get();

return -1;

} //if

PrintMatr(ppMatr, n, m); //печать матрицы из процедуры

WMatrToFile(fNAME\_Out, ppMatr, n, m); //запись матрицы в файл

ReleaseMatr(ppMatr, n); //освободить память

//===========================================================

cin.get(); // ждем

return 0; // выходим из программы

}//end main()

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ВЕКТОР \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**/\*-------------------------------------------------------------\*/**

**/\* Чтение вектора \*/**

**/\*-----------------\*/**

int InputVec(const char \*xfNAME\_VEC, int \*\*xpVec, int \*xM)

{

// \*xpVec - адрес начала вектора

// \*\*xpVec - указатель на адрес начала вектора

int j;

int TotSum = 0;

cout << "\n/1 InputVec: xfNAME\_VEC = " << xfNAME\_VEC << endl;

ifstream fin(fNAME\_VEC); //открытие входного файла

//открытие файла не состоялось

if (!fin)

{

cout << "\a\n\tCannot open file " << fNAME\_VEC << "\n\n";

return 1;

}//if

//читаем размер вектора

fin >> \*xM;

cout << "/2 InputVec: xM = " << \*xM << endl;

//динамическое выделение памяти

\*xpVec = new int[\*xM]; //указатель на массив чисел

for (j = 0; j < \*xM; j++) //читаем вектор

{

fin >> \*((\*xpVec) + j); //(\*xpVec)[j]

cout << "/3 InputVec: \*((\*xpVec) + " << j << ") = "

<< \*((\*xpVec) + j) << " "

<< (\*xpVec)[j] << " " << endl;

TotSum = TotSum + \*((\*xpVec) + j);

} //for j

cout << "\n/2 InputVec: \tTotSum VEC = " << TotSum << " \n";

fin.close(); // закрываем файл исходных данных

return 0;

}//end InputVec()

**/\*-------------------------------------------------------------\*/**

**/\* Печать вектора \*/**

**/\*-----------------\*/**

void PrintVec(int \*pXVec, int Xm)

{

int j;

cout << "\n/PrintVec:\tRECEIVE VECTOR \n";

cout << "/PrintVec: Xm = " << Xm << endl;

for (j = 0; j < Xm; j++) //читаем вектор

cout << " " << \*(pXVec + j);

cout << endl;

} //end PrintVec()

**/\*-------------------------------------------------------------\*/**

**/\* Освободить память \*/**

**/\* вектора \*/**

**/\*--------------------\*/**

void ReleaseVec(int \*xpVec)

{

delete[] xpVec;

}

**// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* МАТРИЦА \*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**/\*-------------------------------------------------------------\*/**

**/\* Чтение матрицы \*/**

**/\*-----------------\*/**

int InputMatr(const char \*xfNAME\_MATR, int \*\*\*xppMatr, int &xN, int &xM)

{

// \*pxVec - адрес начала вектора

int i, j;

int TotSum = 0;

cout << "\n/1 InputMatr: xfNAME\_MATR = " << xfNAME\_MATR << endl;

ifstream fin(fNAME\_MATR); //открытие входного файла

if (!fin)

{

cout << "\a\n\tCannot open file " << fNAME\_MATR << "\n\n";

return 1;

}

//читаем размер матрицы

fin >> xN >> xM;

cout << "/2 InputMatr: xN = " << xN << "\txM = " << xM << endl;

//создаем динамический двумерный массив

\*xppMatr = new int \*[xN]; //указатель на массив из xN диких указателей

//на пустые строки целых чисел неопределенной длины

cout << "\n \*xppMatr = " << \*xppMatr << "\n";

for (i = 0; i<xN; i++)

{

//выделение памяти под каждый элемент строки конкретной длины

(\*xppMatr)[i] = new int[xM]; //указатель на массив целых чисел

cout << "\n (\*xppMatr)[" << i << "] = " << (\*xppMatr)[i] << "\n";

}//for i

cout << "\n/3 InputMatr: \t READ MATRIX \n";

for (i = 0; i<xN; i++) //читаем матрицу

{

for (j = 0; j < xM; j++)

{

/\*

//работает

fin >> \*((\*xppMatr)[i]+j); //считать элемент

cout << \*((\*xppMatr)[i]+j) << " "; //эхо-печать

\*/

/\*

//работает

fin >> \*(\*((\*xppMatr)+i)+j); //считать элемент

cout << \*(\*((\*xppMatr)+i)+j) << " "; //эхо-печать

TotSum += \*(\*((\*xppMatr)+i)+j);

\*/

//работает

fin >> (\*xppMatr)[i][j]; //считать элемент

cout << (\*xppMatr)[i][j] << " "; //эхо-печать

TotSum = TotSum + (\*xppMatr)[i][j];

/\*

// не работает

fin >> \*( ( (\*xppMatr) + i) [j] ); //считать элемент

cout << \*( ( (\*xppMatr) + i) [j]) << " "; //эхо-печать

\*/

}//for j

cout << "\n";

}//for i

cout << "\n/4 InputMatr:\tTotSum MATR = " << TotSum << "\n\n";

for (i = 0; i<xN; i++) //выделенная память

{

cout << "/5 InputMatr: xppMatr[" << i << "] = " << (\*xppMatr)[i] << "\n";

}

return 0;

}//end InputMatr()

**/\*-------------------------------------------------------------\*/**

**/\* Печать матрицы \*/**

**/\*-----------------\*/**

void PrintMatr(int \*\*ppXMatr, int Xn, int Xm)

{

int i, j;

cout << "\n\n/PrintMatr: RECEIVE MATRIX \n\n";

for (i = 0; i<Xn; i++) //читаем матрицу

{

for (j = 0; j < Xm; j++)

{

cout << \*(\*(ppXMatr + i) + j) << " "; //эхо-печать

}//for j

cout << "\n";

}//for i

} //end PrintMatr()

**/\*-------------------------------------------------------------\*/**

**/\* Запись матрицы \*/**

**/\* в файл \*/**

**/\*-----------------\*/**

void WMatrToFile(const char \*xfNAME\_Out\_MATR, int \*\*ppXMatr, int Xn, int Xm)

{

int i, j;

cout << "\n\n/WMatrToFile: WRITE MATRIX TO FILE "

<< fNAME\_Out << "\n\n";

// ofstream fout(fNAME\_Out);

ofstream fout(xfNAME\_Out\_MATR);

//открытие файла не состоялось

if (!fout)

{

cout << "File " << xfNAME\_Out\_MATR

<< "for write NOT open ! \a\n";

return;

}//if

//запись матрицы

for (i = 0; i<Xn; i++)

{

for (j = 0; j < Xm; j++)

{

//записываем четные значения

cout << 2 \* (\*(\*(ppXMatr + i) + j)) << " "; //эхо-печать

fout << 2 \* (\*(\*(ppXMatr + i) + j)) << " "; //в файл

}//for j

cout << "\n";

fout << "\n";

}//for i

fout.close(); //закрыть файл

} //end WMatrToFile()

**/\*-------------------------------------------------------------\*/**

**/\* Освободить память \*/**

**/\* матрицы \*/**

**/\*--------------------\*/**

void ReleaseMatr(int \*\*xppMatr, int xN) //освободить память

{

int i;

printf("\n/1 ReleaseMatr:\t ERASE MEMORY xppMatr\n");

//освобождение памяти в обратном порядке

for (i = 0; i < xN; i++)

{

delete[] xppMatr[i]; //удаляем строки (массивы)

cout << " delete [] xppMatr[" << i << "] = "

<< xppMatr[i] << endl;

// printf ( " delete [] xppMatr[%d] = %p \n"

// , i, xppMatr[i]);

}//for i

delete[] xppMatr; //удаляем массив указателей на строки

cout << " delete [] xppMatr\n";

}//end ReleaseMatr()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of file Ex04\_15.CPP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Пример 4.12** Чтение динамических массивов средствами С

разных размерностей из файла, передача массивов в функцию, запись массива в файл.

На занятии рассмотреть часть примера.

Остальное самостоятельно.

Подготовить в текстовом редакторе файл исходных данных **SRC1.TXT**

4

10 11 12 13

3 5

11 12 13 14 15

21 22 23 24 25

31 32 33 34 35

2 3 4

31 32 33 34

41 42 43 44

51 52 53 54

64 63 62 61

74 73 72 71

84 83 82 81

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*-------------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\Lesson\_03\_Указатели\_Ссылки\Ex04\_09 \*

\* File Name : Ex04\_09.CPP \*

\* Programmer(s) : Чечиков Ю.Б. & Безродных И.П. \*

\* Modifyed By : \*

\* Created : 06/06/05 \*

\* Last Revision : 10/10/05 \*

\* Comment(s) : **ДИНАМИЧЕСКИЕ МНОГОМЕРНЫЕ МАССИВЫ** \*

\* УНИВЕРСАЛЬНАЯ РАБОТА С МАССИВОМ ЧЕРЕЗ УКАЗАТЕЛЬ \*

\* ЧТЕНИЕ РАЗМЕРА ВЕКТОРА И ВЕКТОРА ИЗ ФАЙЛА \*

\* ЧТЕНИЕ РАЗМЕРА МАТРИЦЫ И МАТРИЦЫ ИЗ ФАЙЛА \*

\* ЗАПИСЬ МАТРИЦЫ В ФАЙЛ \*

\* ЧТЕНИЕ РАЗМЕРА КУБА И КУБА ИЗ ФАЙЛА \*

\* ПЕРЕДАЧА ВЕКТОРА, МАТИРИЦЫ И КУБА В ПРОЦЕДУРУ \*

\* ВСЕ ФАЙЛЫ ТЕКСТОВЫЕ \*

\* **ВВОД/ВЫВОД СРЕДСТВАМИ ЯЗЫКА С** \*

\* scanf() работает с **адресами** \*

\* printf() работает со **значениями** \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// читаем данные из файла

#include <stdio.h>

#include <conio.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Г Л О Б А Л Ь Н Ы Е К О Н С Т А Н Т Ы \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const char fNAME[]="SRC1.dat"; //файл исходных данных

const char fNAME\_Out[]="RES1.dat"; //файл результатов

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void PrintVec (int XVec[], int Xm); //(int \*pVec, int Xm);

void PrintMatr (int \*\*ppMatr, int Xn, int Xm);

void PrintCube (int \*\*\*pppCube, int Xk, int Xn, int Xm);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* О С Н О В Н А Я П Р О Г Р А М М А \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

FILE \*fp=NULL; // объявим указатель на входной файл

FILE \*fp\_out=NULL; // объявим указатель на выходной файл

int n; //число строк

int m; //число столбцов

int k; //число плоскостей

int i,j,l;

int TotSum;

fp = fopen (fNAME,"rt"); //открытие входного файла

if (fp == NULL)

{

fprintf(stderr,"\a\n\tCannot open file %s ! \n\n", fNAME);

return 1;

}

**//================================================================**

printf ("\n\t VECTOR \n");

TotSum = 0;

int \*pVec = 0; //указатель на вектор

fscanf (fp,"%d ", &m); //читаем размер вектора

printf ("/1: m = %d \n",m);

**pVec = new int[m];** //указатель на массив **чисел**

for (j = 0; j < m; j++) //читаем вектор

{

fscanf (fp, " %d ", pVec+j);

printf ("/2: \*(pVec +%d) = %4d \n", j, \*(pVec+j));

// printf ("/2: \*(pVec +%d) = %4d %4d %4d %4d \n"

// , j, \*(pVec+j), \*(j+pVec)

// , pVec[j] , [j]pVec );

TotSum = TotSum + \*(pVec+j);

}//for j

printf ("\n\tTotSum VEC = %d \n",TotSum);

PrintVec (pVec, m); //печать вектора из процедуры

//освобождение памяти

delete [] pVec;

**//==================================================================**

printf ("\n\n\t **MATRIX** \n");

TotSum = 0;

//указатель на указатель, который указывает на целое число

int \*\*ppMatr; //двойной **указатель** на массив целых

fscanf (fp," %d %d ",&n, &m); //читаем размер матрицы

printf ("/3: n = %d, m = %d \n", n,m);

ppMatr=new int **\*** [n]; //**указатель на массив диких указателей**

//на n пустых строк неопределенной длины

printf ("\n ppMatr = %p \n", ppMatr);

for (i=0; i<n; i++)

{

//выделение памяти под каждый элемент строки конкретной длины

ppMatr[i]=new int [m]; //указатель на массив **чисел**

printf (" ppMatr[%d] = %p \n", i, ppMatr[i]);

}

printf ("\n\t READ MATRIX \n");

for (i=0; i<n; i++) //читаем матрицу

{

for (j = 0; j < m; j++)

{

/\*

//работает

fscanf (fp, " %d" , ppMatr[i]+j); //считать элемент

printf ("%4d " , \*(ppMatr[i]+j)); //эхо-печать

\*/

/\*

//работает

fscanf (fp, " %d", \*(ppMatr+i)+j );

printf ("%4d " , \*(\*(ppMatr+i)+j));

TotSum += \*(\*(ppMatr+i)+j);

\*/

/\*

//работает

fscanf (fp, " %d", &ppMatr[i][j]); //считать элемент

printf ("%4d " , ppMatr[i][j]); //эхо-печать

TotSum = TotSum + ppMatr[i][j];

\*/

//работает

**fscanf (fp, " %d", &ppMatr[i][j]); //считать элемент**

**printf ("%4d" , ppMatr[i][j]); //эхо-печать**

}//for j

printf ("\n");

}//for i

// printf ("\n\tTotSum MATR = %d \n\n",TotSum);

/\*

for (i=0; i<n; i++) //выделенная память

{

printf (" ppMatr[%d] = %p \n", i, ppMatr[i]);

}

\*/

PrintMatr (ppMatr, n, m);

**//---------------------------------------------------------**

**//запись результатов в файл**

// fp\_out = fopen (fNAME\_Out,"wt");

// if (fp\_out == NULL)

if ((fp\_out = fopen (fNAME\_Out,"wt")) == NULL)

{

printf("\a\n\tCannot open file %s ! \n\n", fNAME\_Out);

return 1;

}

// пишем float матрицу в файл

for (i=0; i<n; i++)

{

for (j = 0; j < m; j++)//форматная запись в файл

fprintf (fp\_out, " %5f", 2\*ppMatr[i][j]+i/10.+j/100.);

fprintf (fp\_out, "\n");

}//for i

fclose(fp\_out); // закрываем выходной файл

**//-----------------------------------------------------------------**

**//освобождение памяти в обратном порядке**

printf ("\n ERASE MEMORY ppMatr\n");

for (i=0; i<n; i++)

{

// printf (" delete [] ppMatr[%d] = %p \n", i, ppMatr[i]);

delete [] ppMatr[i]; //удаляем строки (массивы)

}

delete [] ppMatr; //удаляем массив указателей на строки

**//==================================================================**

**printf ("\n\n\t CUBE \n");**

int **\*\*\***pppCube; //тройной **указатель** на трехмерный массив целых

TotSum = 0;

fscanf (fp," %d %d %d",&k, &n, &m); //читаем размер матрицы

printf ("/5: k = %d, n = %d, m = %d \n", k,n,m);

pppCube = new int **\*\*** [k]; //массив **указателей** на плоскости (матрицы)

// printf ("\n new \*\* pppCube = %p \n", pppCube);

for (l = 0; l<k; l++) //по числу матриц

{

pppCube[l] = new int **\*** [n]; //массив **указателей** на строки

// printf ("\tnew \* pppCube[%d] = %p \n\n", l, pppCube[l]);

for (i=0; i<n; i++) //по числу строк

{

pppCube[l][i] = new int [m]; //указатель на столбец **чисел**

// printf (" \t\tnew \* pppCube[%d][%d] = %p \n"

// , l, i, pppCube[l][i]);

for (j=0; j<m; j++) //по числу столбцов

{

fscanf (fp, "%d ", \*(\*(pppCube + l) + i) +j );

printf ("%4d " , \*(\*(\*(pppCube + l) + i) +j) );

} //for j

printf ("\n");

} //for i

printf ("\n");

} //for l

PrintCube (pppCube, k, n, m);

**//---------------------------------------------------------**

**//освобождение памяти в обратном порядке**

printf ("\n ERASE MEMORY pppCube\n");

for (l=0; l<k; l++) //по числу матриц

{

for (i=0; i<n; i++) //по числу строк

{

//printf (" delete [] pppCube[%d][%d] = %p \n", l, i, pppCube[l][i]);

**delete [] pppCube[l][i]; //удаляем строки (массивы)**

}

// printf (" delete [] pppCube[%d] = %p \n\n", l, pppCube[l]);

**delete [] pppCube[l];** //указатель на плоскости

}

// printf ("\n delete [] \*\* pppCube = %p \n", pppCube);

**delete [] pppCube;** //удаляем массив указателей на плоскости

// printf ("\n");

**//==================================================================**

fclose(fp); // закрываем файл исходных данных

getch(); // ждем

return 0; // выходим из программы

}//end main()

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

/\*-----------------------------------------------------------\*/

/\* Печать вектора \*/

/\*-----------------\*/

**void PrintVec (int \*pXVec, int Xm)**

{

int j;

printf ("\n/PrintVec:\tRECEIVE VECTOR \n");

printf ("/PrintVec: Xm = %d \n",Xm);

for (j = 0; j < Xm; j++) //читаем вектор

printf (" %d ", \*(pXVec+j));

printf ("\n");

} //end PrintVec()

/\*-----------------------------------------------------------\*/

/\* Печать матрицы \*/

/\*-----------------\*/

**void PrintMatr (int \*\*ppXMatr, int Xn, int Xm)**

{

int i, j;

printf ("\n\n/PrintMatr: RECEIVE MATRIX \n\n");

for (i=0; i<Xn; i++) //читаем матрицу

{

for (j = 0; j < Xm; j++)

{

printf ("%4d", \*(\*(ppXMatr+i)+j)); //эхо-печать

}//for j

printf ("\n");

}//for i

} //end PrintMatr()

/\*-----------------------------------------------------------\*/

/\* Печать куба \*/

/\*-------------\*/

**void PrintCube (int \*\*\*pppXCube, int Xk, int Xn, int Xm)**

{

int i,j,l;

printf ("\n\n/PrintCube: RECEIVE CUBE \n\n");

for (l=0; l<Xk; l++) //число плоскостей

//читаем матрицу

{

for (i = 0; i < Xn; i++)

{

for (j = 0; j < Xm; j++)

{

printf ("%4d", pppXCube[l][i][j]); //эхо-печать

}//for j

printf ("\n");

}// for i

printf ("\n");

}//for l

} //end PrintCube()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of file Ex04\_09.CPP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**НА ДОМ**: разобрать пример с чтением из файла и заполнением **статических** массивов разных размерностей.

**Пример 4.13** Чтение **статических** массивов

разных размерностей из файла, передача массивов в функцию, запись массива в файл.

File Name :Ex04\_10\_стат\_масс\_файл.cpp

Подготовить в текстовом редакторе файл исходных данных **SRC1.TXT**

4

10 11 12 13

3 5

11 12 13 14 15

21 22 23 24 25

31 32 33 34 35

2 3 4

31 32 33 34

41 42 43 44

51 52 53 54

64 63 62 61

74 73 72 71

84 83 82 81

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* К У Р С Ы С + + \*

\*-------------------------------------------------------------------\*

\* Project Type : Win32 Console Application \*

\* Project Name : \Курсы\_С++\1 КУРС\MATRIX\ Ex04\_10 \*

\* File Name : Ex04\_10\_стат\_масс\_файл.CPP \*

\* Programmer(s) : Чечиков Ю.Б. & Безродных И.П. \*

\* Modifyed By : \*

\* Created : 11/03/05 \*

\* Last Revision : 10/10/05 \*

\* Comment(s) : **МАССИВЫ СТАТИЧЕСКИЕ \***

**\***  ЧТЕНИЕ РАЗМЕРА ВЕКТОРА И ВЕКТОРА ИЗ ФАЙЛА \*

\* ЧТЕНИЕ РАЗМЕРА МАТРИЦЫ И МАТРИЦЫ ИЗ ФАЙЛА \*

\* ЧТЕНИЕ РАЗМЕРА КУБА И КУБА ИЗ ФАЙЛА \*

\* ЗАПИСЬ МАТРИЦЫ В ФАЙЛ \*

\* УНИВЕРСАЛЬНАЯ РАБОТА С МАССИВОМ ЧЕРЕЗ УКАЗАТЕЛЬ \*

\* ПЕРЕДАЧА ВЕКТОРА, МАТИРИЦЫ И КУБА В ПРОЦЕДУРУ \*

\* ВСЕ ФАЙЛЫ ТЕКСТОВЫЕ \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// читаем данные из айла

#include <stdio.h>

#include <conio.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Г Л О Б А Л Ь Н Ы Е К О Н С Т А Н Т Ы \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const N\_MAX = 10; //максимальное число строк в матрице

const M\_MAX = 10; //максимальное число столбцов в строке (векторе)

const K\_MAX = 10; //максимальное число матриц в кубе

const char fNAME[]="SRC1.dat"; //файл исходных данных

const char fNAME\_Out[]="RES1.dat"; //файл результатов

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void PrintVec (int XVec[], int Xm); //через индексы

void PrintMatr (int XMatr[][M\_MAX], int Xn, int Xm); //через индексы

void PrintMatrP (int \*pXM, int Xn, int Xm); //через указатель

void PrintCube (int XCube[][N\_MAX][M\_MAX], int Xk, int Xn, int Xm);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Г Л О Б А Л Ь Н Ы Е П Е Р Е М Е Н Н Ы Е \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* О С Н О В Н А Я П Р О Г Р А М М А \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main()

{

FILE \*fp=NULL; // объявим указатель на входной файл

FILE \*fp\_out=NULL; // объявим указатель на выходной файл

int n; //число строк

int m; //число столбцов

int k; //число плоскостей

int AVec[M\_MAX]; //вектор

int AMatr[N\_MAX][M\_MAX]; //матрица

int ACube[N\_MAX][M\_MAX][K\_MAX]; //куб

int i,j,l;

int TotSum; //сумма элементов массива

fp = fopen (fNAME,"rt"); //открытие входного файла

if (fp == NULL)

{

fprintf(stderr,"\a\n\tCannot open file %s ! \n\n", fNAME);

return 1;

}

**//================================================================**

**printf ("\n\t VECTOR \n");**

TotSum = 0; //сумма элементов массива

fscanf (fp,"%d ", &m); //читаем размер вектора

printf ("/1: m = %d \n",m); //эхо-печать

//читаем вектор

for (j = 0; j < m; j++) //по числу столбцов вектор-строки

{

fscanf (fp, " %d ", &AVec[j]);

printf ("/2: AVec[%d] = %d \n", j, AVec[j]);

TotSum = TotSum + AVec[j];

}//for j

printf ("\n\tTotSum VEC = %d \n",TotSum); //заголовок

PrintVec (AVec, m); //печать вектора из процедуры

**//==================================================================**

**printf ("\n\n\t MATRIX \n");**

TotSum = 0; //сумма элементов массива

fscanf (fp," %d %d ",&n, &m); //читаем размер матрицы

printf ("/3: n = %d, m = %d \n", n,m); //эхо-печать

//читаем матрицу

for (i=0; i<n; i++) //по числу строк

{

for (j = 0; j < m; j++) //по числу столбцов

{

fscanf (fp, " %d", &AMatr[i][j]); //считать элемент

printf ("%4d", AMatr[i][j]); //эхо-печать

TotSum = TotSum + AMatr[i][j];

}//for j

printf ("\n");

}//for i

printf ("\n\tTotSum MATR = %d \n",TotSum);

PrintMatr (AMatr, n, m);

// PrintMatrP (AMatr, n, m); //можно через указатель

**//-----------------------------------------------------------------**

**// УНИВЕРСАЛЬНАЯ РАБОТА С МАССИВОМ ЧЕРЕЗ УКАЗАТЕЛЬ**

int \*pint; //указатель на целое число

int Sum3 = 0; //сумма элементов матрицы, кратных 3

//инициализация указателя

pint = &AMatr[0][0]; //адресом первого элемента

// pint = AMatr; //адресом начала массива

printf ("/Pointer AMatr \n");

//читаем матрицу

for (i=0; i<n; i++) //по числу строк

{

for (j = 0; j < m; j++) //по числу столбцов

{

if ((\*(pint + i\*M\_MAX + j)) % 3 == 0) //числа, кратные 3

{

// Sum3 += AMatr[i][j]; //через индекс

Sum3 += \*(pint + i\*M\_MAX + j); //через указатель

//эхо-печать чисел, кратных 3

printf ("%3d %3d %4d %4d \n",i, j, AMatr[i][j], \*(pint + i\*M\_MAX + j));

}//if

//эхо-печать всех чисел

// printf ("%3d %3d %4d %4d \n",i, j, AMatr[i][j], \*(pint + i\*M\_MAX + j));

}//for j

// printf ("\n");

}//for i

printf ("\n\tSum3 MATR = %d \n",Sum3);

getch(); //бряк по клавише без эха на экране

**//-----------------------------------------------------------**

**// int (\*pVec)[M\_MAX]; //указатель на массив из M\_MAX элементов**

/\*

int \*\*pAM; //указатель на указатель, указывающий на целое

int Sum3p = 0; //сумма элементов матрицы, кратных 3

\*pAM = &AMatr[0]; //указатель на массив строк

printf ("/Pointer AMatr \n");

//читаем матрицу

for (i=0; i<n; i++) //по числу строк

{

for (j = 0; j < m; j++) //по числу столбцов

{

if ((\*(pint + i\*M\_MAX + j)) % 3 == 0) //числа, кратные 3

{

// Sum3 += AMatr[i][j]; //через индекс

Sum3 += \*(pint + i\*M\_MAX + j); //через указатель

//эхо-печать чисел, кратных 3

printf ("%3d %3d %4d %4d \n",i, j, AMatr[i][j]

, \*(pint + i\*M\_MAX + j));

}//if

//эхо-печать всех чисел

// printf ("%3d %3d %4d %4d \n",i, j, AMatr[i][j]

, \*(pint + i\*M\_MAX + j));

}//for j

// printf ("\n");

}//for i

printf ("\n\tSum3 MATR = %d \n",Sum3);

getch(); //бряк по клавише без эха на экране

\*/

**//==================================================================**

**//работа с трехмерным массивом**

printf ("\n\n\t CUBE \n");

TotSum = 0; //сумма элементов массива

fscanf (fp," %d %d %d",&k, &n, &m); //читаем размер матрицы

printf ("/5: k = %d, n = %d, m = %d \n", k,n,m);

for (l=0; l<k; l++) //по числу плоскостей

//читаем матрицу

{

for (i = 0; i < n; i++) //по числу строк

{

for (j = 0; j < m; j++) //по числу столбцов

{

fscanf (fp,"%d",&ACube[l][i][j]); //через индекс

printf ("%4d", ACube[l][i][j]); //эхо-печать

TotSum = TotSum + ACube[l][i][j];

}//for j

printf ("\n");

}// for i

printf ("\n");

}//for l

printf ("\n\tTotSum CUBE = %d \n",TotSum);

PrintCube (ACube, k, n, m);

//==================================================================

fclose(fp); // закрываем файл

**//==================================================================**

**//запись результатов в файл**

// fp\_out = fopen (fNAME\_Out,"wt");

// if (fp\_out == NULL)

if ((fp\_out = fopen (fNAME\_Out,"wt")) == NULL)

{

fprintf(stderr,"\a\n\tCannot open file %s ! \n\n", fNAME\_Out);

return 1;

}

// пишем матрицу в файл

for (i=0; i<n; i++) //по числу строк

{

for (j = 0; j < m; j++) //по числу столбцов

{

//форматная запись в файл

fprintf (fp\_out, " %5d", **2\*AMatr[i][j]**); //поменяли значения

}//for j

fprintf (fp\_out, "\n");

}//for i

fclose(fp\_out); // закрываем выходной файл

getch(); // ждем бряк по клавише без эха на экране

return 0; // выходим из программы

} //main

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Р Е А Л И З А Ц И Я Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**/\*---------------------------------------------------------------------\*/**

**/\* Печать вектора \*/**

**/\*-----------------\*/**

void PrintVec (int XVec[], int Xm)

{

int j;

printf ("\n/PrintVec:\tRECEIVE VECTOR \n");

printf ("/PrintVec: Xm = %d \n",Xm);

//читаем вектор

for (j = 0; j < Xm; j++) //по числу столбцов вектор-строки

{

printf (" %d ", XVec[j]); //через индекс

}//for j

printf ("\n");

}

**/\*---------------------------------------------------------------------\*/**

**/\* Печать матрицы \*/**

**/\* через индексы \*/**

**/\*-----------------\*/**

void PrintMatr (int XMatr[][M\_MAX], int Xn, int Xm)

{

int i, j;

printf ("\n\n/PrintMatr: RECEIVE MATRIX \n\n");

//читаем матрицу

for (i=0; i<Xn; i++) //по числу строк

{

for (j = 0; j < Xm; j++) //по числу столбцов

{

printf ("%4d", XMatr[i][j]); //через индекс

}//for j

printf ("\n");

}//for i

}

**/\*---------------------------------------------------------------------\*/**

**/\* Печать матрицы \*/**

**/\* через указатель \*/**

**/\*-----------------\*/**

void PrintMatrP (int \*pXM, int Xn, int Xm)

{

int i, j;

printf ("\n\n/PrintMatrP: RECEIVE MATRIX \n\n");

//читаем матрицу

for (i=0; i<Xn; i++) //по числу строк

{

for (j = 0; j < Xm; j++) //по числу столбцов

{

printf ("%4d", pXM+i\*M\_MAX+j); //через указатель

}//for j

printf ("\n");

}//for i

}

**/\*---------------------------------------------------------------------\*/**

**/\* Печать куба \*/**

**/\*-------------\*/**

void PrintCube (int XCube[][N\_MAX][M\_MAX], int Xk, int Xn, int Xm)

{

int i,j,l;

printf ("\n\n/PrintMatr: RECEIVE CUBE \n\n");

for (l=0; l<Xk; l++) //по числу плоскостей

{

for (i = 0; i < Xn; i++) //по числу строк

{

for (j = 0; j < Xm; j++) //по числу столбцов

{

printf ("%4d", XCube[l][i][j]); //через индекс

}//for j

printf ("\n");

}// for i

printf ("\n");

}//for l

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of file Ex04\_10.CPP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**КОНЕЦ**

Утро было прохладное. В жемчужном небе путалось бледное солнце. В травах кричала мелкая птичья сволочь.

Дорожные птички "пастушки" медленно переходили дорогу перед самыми колесами автомобиля. Степные горизонты источали такие бодрые запахи, что, будь на месте Остапа какой-нибудь крестьянский писатель-середнячок из группы "Стальное вымя", не удержался бы он, вышел бы из машины, сел бы в траву и тут же на месте начал бы писать на листах походного блокнота новую повесть, начинающуюся словами: "Инда взопрели озимые. Рассупонилось солнышко, расталдыкнуло свои лучи по белу светушку. Понюхал старик Ромуальдыч свою портянку и аж заколдобился... "

Но Остап и его спутники были далеки от поэтических восприятий. Вот уже сутки они мчались впереди автопробега.