**Лабораторная работа №5**

Запись/ чтение массива структур в файл.

**Цель:** Изучить структурные шаблоны, структурные переменные, массивы структур, вложенные структуры и указатели на структуры, научиться инициализировать структуры, передавать информацию о структурах функциям.

# МАССИВЫ СТРУКТУР

## Описание массива структур

Процесс описания массива структур совершенно аналогичен описанию любого другого типа массива. Чтобы объявить массив структур, необходимо сначала определить структуру, а затем объявить массив элементов этого структурного типа:

*struct stud my[5];*

Этот оператор объявляет ***my*** массивом, состоящим из 5 элементов. Каждый элемент массива представляет собой структуру типа *stud*. Таким образом, my[0] является первой *stud*-структурой, my[1] – второй *stud*-структурой и т.д. Имя ***my*** само по себе не является именем структуры; это имя массива, содержащего структуры. Работа с массивами структур была приведена в примере 2 стр. 9-10.

## Определение элементов массива структур

Чтобы получить доступ к конкретной структуре в массиве структур, необходимо индексировать имя структуры. Например, чтобы отобразить на экране содержимое члена *bal* третьей структуры, используют следующую инструкцию:

***cout << my[2].bal;***

Подобно всем переменным массивов, у массивов структур индексирование начинается с нуля. При определении элементов массива структур применяются те же правила, что используются для отдельных структур: сопровождаем имя структуры [индекс] операцией получения элемента «•» и именем элемента:

*my[0].name*, где name – первый элемент массива.

*my[4].name*, где name – пятый элемент массива.

***Пример 4***: Изменим пример 3, создав массив структур stud, инициализируем его и выведем на экран полученные данные.

#include <iostream>

using namespace std;

struct stud {

char name[20];

char group\_name[20];

float bal;

};

void main()

{

setlocale(LC\_ALL, "Russian");

struct stud my[3] = { "Липницкая И.А.", "УИР 2", 9.85,

"Скоморощенко К.С.", "УИР 1", 9.25,

"Щурок Е.А.", "УИР 3", 8.25,

};

for (int i = 0; i < 3; i++)

{

cout << "ФИО : ";

cout.width(20);

cout << left << my**[i].**name << '\t'; /\* правильно \*/

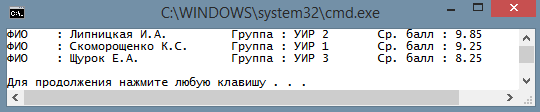
cout << "Группа : " << my[i].group\_name << '\t';

cout << " Ср. балл : " << my[i]. bal << endl;

}

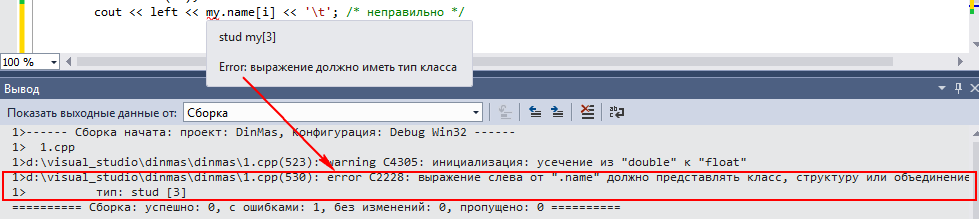
cout << endl;

}



Индекс массива присоединяется к имени структуры ***my***, а не к концу элемента:

cout << left << my.name**[i]** << '\t'; /\* неправильно \*/



Используем ***my[i].name*** для обращения к члену структуры *name*, так как ***my[i]*** – имя структурной переменной.

Выведем на печать 4-й элемент поля *name* структурной переменной *my[i]:*

struct stud my[3] = { "Липницкая И.А.", "УИР 2", 9.85,

"Скоморощенко К.С.", "УИР 1", 9.25,

"Щурок Е.А.", "УИР 3", 8.25,

};

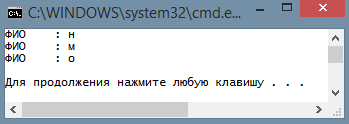
for (int i = 0; i < 3; i++)

{

cout << "ФИО : ";

cout << **my[i].name[3]** << endl;

}



Это будет четвертый символ строкового элемента *name* (*name [3]*) структуры *my[i]* типа stud.

***Пример 5***: Объявление и использование массива структур как описыние точки на плоскости через систему координат.

#include <iostream>

#include <stdio.h>

using namespace std;

struct Point // создается шаблон структур типа Point

{

int x;

int y;

};

void main(void)

{

Point m[10]; // объявляется 10 структур типа Point,

// для которых выделяется память

for (int i = 0; i < 10; i++) // заполнение структур значениями

{

m[i].x = i\*(i - 2);

m[i].y = i \* 3;

cout << " x = " << m[i].x << '\t' << " y = " << m[i].y << endl;

}

cout << endl;

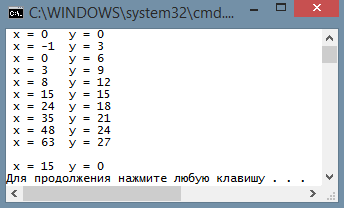
int d;

d = m[5].x; // d = 15

d = m[0].y; // d = 0

cout << " x = " << m[5].x << '\t' << " y = " << m[0].y << endl;

}



***Пример 6:*** Объявление и использование массива указателей на структуры. Дан массив указателей на 10 структур *Point.*

#include <iostream>

#include <stdio.h>

using namespace std;

struct Point

{

int x;

int y;

};

void main(void)

{

Point \* pm[10];

for (int i = 0; i < 10; i++) // выделение памяти для каждой структуры

{

pm[i] = new Point;

}

for (int i = 0; i < 10; i++) // заполнение структур произвольными значениями

{

pm[i]->x = i \* 5 + 2;

pm[i]->y = i\*i\*i;

cout << " x = " << pm[i]->x << '\t' << " y = " << pm[i]->y << endl;

}

cout << endl;

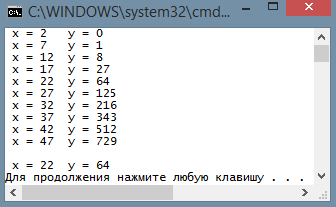
int d; // проверка

d = pm[4]->x; // d = 22

d = pm[4]->y; // d = 64

cout << " x = " << pm[4]->x << '\t' << " y = " << pm[4]->y << endl;

}



***Обобщение:*** индексы *[j]ё*, находящиеся справа от операции «•», относятся к отдельным элементам, в то время как индексы *[i]*, расположенные слева от операции, относятся к массивам структур (*my[i].name[j]*)*.*

## Вложенные структуры

Шаблон любой структуры может включать в себя другие структуры. Структура, являющаяся компонентом другой структуры, называется ***вложенной***. Экземпляр такой структуры был рассмотрен в примере 2 стр. 9-10.

***Для размещения вложенной структуры в шаблон*** она просто описывается точно так же, как это делалось бы для переменной типа *int*.

Например, есть шаблон структуры *Point*, который описывает точку на плоскости:

*struct Point*

*{*

*int x;*

*int y;*

*};*

Создадим шаблон *Triangle*, описывающий треугольник на плоскости:

*struct Triangle*

*{*

*Point p1; // вложенная структура*

*Point p2; // вложенная структура*

*Point p3; // вложенная структура*

*char comment[50]; // комментарий к фигуре*

*} tp;*

В шаблоне *Triangle* описывается три вложенных структуры (точки) *p1, p2, p3*, являющиеся переменнными типа *struct Point*.

**Получение доступа к элементу вложенной структуры**

Нужно дважды использовать операцию точка «•»:

tr**.**p1**.**x = 25;

tr**.**p2**.**x = 10;

tr**.**p3**.**x = 72;

Интерпретируем эту конструкцию, перемещаясь слева направо: (*tr.p1*).*x* –первым находим элемент *tr*, далее элемент *p1* структуры *tr*, а затем его элемент *x*.

#include <iostream>

using namespace std;

struct Point

{

int x;

int y;

};

struct Triangle

{

Point p1; // вложенная структура

Point p2; // вложенная структура

Point p3; // вложенная структура

char comment[50]; // комментарий к фигуре

} tr[] = {

{ 25, 45 },

{ 10, 86 },

{ 72, 48 },

"Координаты точек"

}**;**

void main(void)

{

setlocale(LC\_ALL, "Russian");

cout << tr.comment <<":" << endl;

cout << " x = " << tr.p1.x << '\t' << " y = " << tr.p1.y << endl;

cout << " x = " << tr.p2.x << '\t' << " y = " << tr.p2.y << endl;

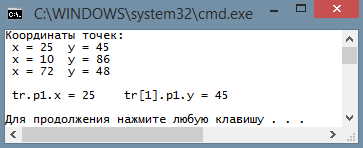
cout << " x = " << tr.p3.x << '\t' << " y = " << tr.p3.y << endl;

cout << endl;

// проверка

cout << " tr.p1.x = " << tr.p1.x << '\t' << " tr[1].p1.y = " << tr.p1.y << endl << endl;

}



# ПЕРЕДАЧА СТРУКТУР В ФУНКЦИЮ

**Структура** является **типом данных** созданным пользователем, к ней можно применять такие же операции, как и к встроенным типам. Рассмотрим работу структур (**struct**) с функциями.

1. Структуру можно передавать в функцию целиком:

#include <iostream>

#include <stdio.h>

using namespace std;

struct stud {

char name[20];

char group\_name[20];

float bal;

} **my[]** ={ "Липницкая И.А.", "УИР 2", 9.85,

"Скоморощенко К.С.", "УИР 1", 9.25,

"Щурок Е.А.", "УИР 3", 8.25,

}**;**

void print\_stud(**struct stud str**); //Прототип функции вывода

void main()

{

setlocale(LC\_ALL, "Russian");

for (int i = 0; i < 3; i++)

{

print\_stud(my[i]);

}

}

void print\_stud(struct stud str)

{

printf("ФИО : %15s\t", str.name);

printf("Группа : %s \t", str.group\_name);

printf(" Ср. балл : %.2f\n", str.bal);

}

1. Можно передать указатель на структуру.

void print\_stud(struct stud\* str); //Прототип функции вывода

void vvod\_stud(struct stud\* str, int n); //Прототип функции ввода

Код программы:

#include <iostream>

#include <stdio.h>

using namespace std;

struct stud {

char name[20];

char group\_name[20];

float bal;

};

void vvod\_stud(struct stud\* str, int n); // Размерность массива int n

void print\_stud(struct stud\* str);

void main()

{

setlocale(LC\_ALL, "Russian");

stud \*pmy = new stud [3] ;

vvod\_stud(pmy, 3);

/\*}\*/

print\_stud(pmy);

cout << endl;

}

void print\_stud(struct stud\* str)

{

for (int i = 0; i < 3; i++)

{

printf("ФИО : %15s\t", str->name);

printf("Группа : %s \t", str->group\_name);

printf(" Ср. балл : %.2f\n", str->bal);

str++;

}

}

void vvod\_stud(struct stud\* str, int n)

{

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

{

printf("\nВведите \n");

printf("ФИО : ");

gets(str[i].name);

printf("Группа : ");

gets(str[i].group\_name);

printf("Ср. балл : ");

cin >> str[i].bal;

cin.ignore();

}

}

1. Можно передать элементы структуры по отдельности:

void stud\_out(char name[20], float f)

{

printf("имя: %s; балл: %.1f\n", name, f);

}

void main()

{

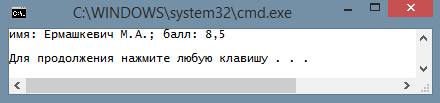
setlocale(LC\_ALL, "Russian");

struct stud a = { "Ермашкевич М.А.", "УИР 2", 8.5 };

stud\_out(a.name, a.bal);

cout << endl;

}



**Повторение:** При передаче самой структуры или ее элементов передаются значения. Т.е. при изменении элементов структуры в функции, настоящие элементы останутся нетронутыми. Для того, чтобы передавались настоящие элементы, в функцию надо передавать структуру или с помощью указателей, или с помощью ссылок.

***Пример:*** Создать структуру *Студент*: курс, группа, ФИО, какие предметы проходят, оценки за экзамен. Объявить динамический массив структур и заполнить его. Вывести на экран ведомость успеваемости. Передать структуру в функцию (ввод данных в массив структур).

Когда пользователь вводит данные в ответ на операцию извлечения, то эти данные помещаются в буфер ***std::cin***. Буфер данных – это просто часть памяти, зарезервированная для временного хранения данных, когда они перемещаются из одного места в другое. В этом случае буфер используется для хранения пользовательского ввода, пока он находится в режиме ожидании выделения для него переменных. В некоторых случаях возникает необходимость пропустить часть символов строки от начала до достижения конца строки (EOL) или конца файла (EOF). Именно этому и отвечает функция ***ignore()***. Она принимает два параметра: число пропускаемых символов и символ разделения. Например, вызов функции ignore(80, '\n') приведет к пропуску 80 символов, если ранее не будет найден символ начала новой строки. Последний затем будет удален из буфера, после чего функция ignore() завершит свою работу.

#include <stdlib.h>

#include <iostream>

#include <Windows.h>

using namespace std;

struct Stud1 //Объявление шаблона структуры Stud1

{

int Kurs;

int Gruppa;

char FIO[20];

char Predmet[30];

int Ocenka;

};

void GetData(Stud1 \*M, int N); //Прототип функции

void main() //Главная функция

{

setlocale(LC\_ALL, "Russian"); //Работа с кириллицей

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("CLS"); //Очистка экрана

int i = 0;

int N;

cout << "Введите N: ";

cin >> N;

Stud1 \*M1 = new Stud1[N]; //Объявление динамического массива

// из N структур типа Stud1

GetData(M1, N); //Ввод данных в массив структур

cout << "\n";

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

{

if (M1[i].Ocenka <= 3)

{

cout << "Студент " << M1[i].FIO << " не сдал экзамен по предмету " << M1[i].Predmet << " : " << M1[i].Ocenka << endl << endl;

}

}

delete[]M1;

system("PAUSE");

}

void GetData(Stud1 \*M, int N) //Функция заполняет данными структуру Stud1

{

cin.ignore(); // Удаляем лишние значения

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

{

setlocale(LC\_ALL, "Russian");

cout << "\n";

cout << "Курс: ";

cin >> M[i].Kurs;

cout << "Группа: ";

cin >> M[i].Gruppa;

cin.ignore();

cout << "ФИО: ";

cin.getline(M[i].FIO, 20);

//gets(M[i].FIO); //Второй вариант ввода данных

cout << "Предмет: ";

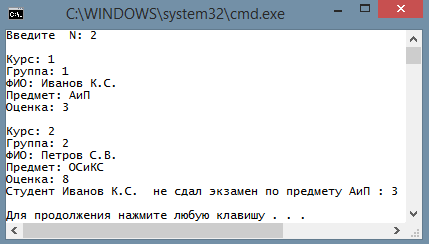
cin.getline(M[i].Predmet, 30);

cout << "Оценка: ";

cin >> M[i].Ocenka;

}

}



### ЗАДАНИЕ

Определить шаблон структуры, содержащий информацию о вашей семье (родители, сестры, братья) и их анкетные данные (дата рождения, пол, место рождения). Используйте вложенные структуры. Выполнить заполнение структуры данными через функцию. Организовать поиск информации и вывод результатов. Например, год рождения вашей сестры, общее количество лет вашей семье, кто самый младший в семье, количество мужчин и женщин в семье и т.д.