**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В. И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

### Тема: **Структуры**

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. |  | Чванов М.В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цели работы**: изучение и организация структур; получение практических навыков работы со структурами; определение преимуществ и недостатков использования структур.

**Основные теоретические положения:**

### Определение структур

Структуры представляют собой группы связанных между собой, как правило, разнотипных переменных, объединенных в единый объект, в отличие от массива, все элементы которого однотипны. В языке C++ структура является видом класса и обладает всеми его свойствами.  
Класс - в ООП представляет собой шаблон для создания объектов, обеспечивающий начальные значения состояний: инициализация полей-переменных и реализация поведения функций или методов. ООП в данном курсе не рассматривается, но иногда будут упоминаться понятия, связанные с ним.

Для определения структуры применяется ключевое слово **struct**, а сам формат определения выглядит следующим образом:

struct [имя\_типа] {

тип\_1 элемент\_1;

тип \_2 элемент\_2;

…

тип\_k элемент\_k;

} [ список\_описателей ];

Каждая входящая в структуру переменная называется членом (полем, элементом) структуры и описывается типом данных и именем. Поля структуры могут быть любого типа данных. Их количество не лимитировано.

Вся эта конструкция является инструкцией языка программирования, поэтому после нее всегда должен ставиться символ ‘;’.

struct Student {

int age;

char name[10];

int group;

};

После определения структуры можно её использовать. Для начала можно определить объект структуры - по сути обычную переменную, которая будет представлять описанную ранее структуру.

struct Student ivan;

Здесь определена переменная ivan, которая представляет структуру **Student**.

При описании структуры память для размещения данных не выделяется. Работать с описанной структурой можно только после того, как будет определена переменная (переменные) этого типа данных, только при этом компилятор выделит необходимую память.

 Еще один способ определения структуры представляет ключевое слово **typedef**:

typedef struct {

int age;

char name[10];

} Person;

В конце определения структуры после закрывающей фигурной скобки идет ее обозначение - в данном случае person. В дальнейшем можно использовать это обозначение для создания переменной структуры.

Person tom = {23, "Tom"};

Еще один способ определить структуру представляет применение препроцессорной директивы **#define**:

#define PERSON struct {int age; char name[20];}

int main(void) {

PERSON tom = {23, "Tom"};

return 0;

}

В данном случае директива define определяет константу PERSON, вместо которой при обработке исходного кода препроцессором будет вставляться код структуры struct {int age; char name[20];}

В языке С++ отличие от С при определении переменной не нужно использовать слово **struct**.

### Инициализация структур

Инициализация структур аналогична инициализации массивов: в фигурных скобках передаются значения для элементов структуры по порядку. Так как в структуре student первым определено свойство, которое представляет тип **int**- число, то в фигурных скобках вначале идет число. И так далее для всех элементов структуры по порядку.

При определении переменной структуры ее можно сразу инициализировать, присвоив какое-нибудь значение:

struct Student ivan = {23, "Ivan", 0393};

Инициализировать структуру можно сразу после её объявления. Для инициализации структуры значения ее элементов перечисляют в фигурных скобках в порядке их описания:

struct Complex {

float real, im;

} data [2][2] = {{{1,1}, {2,2}}, {{3,3}, {4,4}}};

Все поля структурных переменных располагаются в непрерывной области памяти одно за другим. Общий объем памяти, занимаемый структурой, равен сумме размеров всех полей структуры. Для определения размера структуры следует использовать инструкцию **sizeof()**.

**Вывод:** в ходе лабораторной работы я больше узнал о структурах и научился их использовать на практике.