**Практическое занятие №11**

**"Структуры и объединения"**

**Теория**

## Общие сведения о структурах

Структура представляет собой совокупность полей, принадлежащих к различным типам. Поля структуры могут иметь любой тип, кроме типа этой структуры, но могут быть указателями на него. Для объявления структур используется ключевое слово struct, после которого указывается имя типа (структуры), за которым в фигурных скобках следуют объявления типа каждого поля:

struct [имя типа]

{

тип1 поле1;

тип2 поле2;

...

типN полеN;

};

Например, структура, представляющая собой точку на плоскости, задаваемую двумя координатами, может быть объявлена следующим образом:

struct point

{

float x;

float y;

};

или

struct point

{

float x,y;

};

Поля структуры называют также членами, элементами. С помощью структур обычно описывают сложные данные, характеризующиеся несколькими значениями различных типов. Например, сотрудник может описываться своими фамилией, именем, отчеством, полом, годом рождения, окладом. Студент характеризуется также фамилией, именем, отчеством, полом, годом рождения, а также названием группы, оценками. В этих случаях удобно характеризовать каждый объект не совокупностью отдельных переменных, а одной переменной, имеющей несколько полей, причем эти поля рассматриваются как одно целое, а не как разрозненный набор элементов (переменных).

В программе поля структуры могут иметь те же имена, что и другие переменные и поля в других структурах. В разных структурах можно иметь одинаковые названия полей. Однако это может приводить к ошибкам (можно легко перепутать поля разных структур), поэтому такой подход не соответствует принципам хорошего стиля программирования.

Использование имени структуры при ее объявлении фактически означает объявление нового типа данных. Имя структуры можно использовать далее как имя типа, например, point a,b; Можно не вводить имя типа, а после объявления структуры указать имена переменных, например,

struct point

{

float x,y;

} a,b;

Имя типа структуры можно использовать сразу после объявления, а её определение дать позже. Так обычно поступают при объявлении типа структуры, являющейся элементом списка:

struct tElem; //объявление

struct tUkaz

{

tElem \*p;

tUkaz \*right, \*left;

};

struct tElem //определение

{

int nom;

char fam[30];

};

Для инициализации структур значения ее элементов записывают в фигурных скобках в порядке их описания:

struct

{

char fam[30];

int god;

float doch;

} person = {“Петров”, 1958, 1050.56};

При инициализации массива структур следует заключать в фигурные скобки каждый элемент массива, принимая во внимание, что многомерный массив – это массив, состоящий из массивов:

struct mpoint

{

float x,y;

} mas[2][3]={

{{2.2,-4.4}, {4.5, -9.8}, {7.7, -6.6}},

{{1.1,0.7}, {34.6,-12.5}, {9.9,-0.5}}

};

Для переменных одного и того же типа определена операция присваивания, при выполнении которой происходит поэлементное копирование полей одной структуры в другую структуру. Например, если объявлены переменные point p1, p2; то можно записать оператор присваивания p2=p1;. Операции над структурой выполняются для отдельных полей, т.е. проводится поэлементная обработка.

Структуру можно передавать в функцию и возвращать из функции в качестве результата.

При работе со структурами следует принимать во внимание, что размер памяти, занимаемый структурой, может не равняться сумме размеров ее элементов, т.к. поля могут выравниваться по границе слов.

Доступ к полям структуры осуществляется с помощью операции выбора '.' (точка), если работа ведется непосредственно с переменной, и с помощью той же операции, но обозначаемой в виде стрелки -> (парой знаков минус и больше), если используется указатель на структуру. Например,

point a, \*b;

a.x=4.4; a.y=-9.7;

b->x=34.5; b->y=8.6;

Элементом структуры может быть другая структура, объявленная ранее (с другим именем типа), или определённая внутри данной (с именем типа или без). В этом случае доступ к ее элементам выполняется с помощью двух операций выбора:

struct S1

{

int i;

float r;

};

struct S2

{

S1 z; //поле типа структура S1

float c;

};

S2 y[2];

y[0].z.i=2;//поле i структуры типа S1

y[0].z.r=2.2; //поле r структуры типа S1

y[0].c=8.02;

y[1].z.i=6;//поле i структуры типа S1

y[1].z.r=-12.3; //поле r структуры типа S1

y[1].c=1.75;

Поля разных структур могут иметь одинаковые имена, в этом случае никаких проблем не возникает, поскольку они имеют разные области видимости. Вне области видимости полей структуры использованные в нём имена могут использоваться и для других объектов. Например, в следующем фрагменте программы имя q используется и как имя поля, и как переменная определяемой структуры, а имя дано имени поля и простой переменной типа float.

struct {float q,y;} q;

float y;

q.q=5.5f; q.y=-90.9f; y=10.1f;

printf("a.a=%5.1f a.y=%5.1f y=%5.1f",q.q,q.y,y);

## Объединения

Объединение представляет собой частный случай структуры, все поля которой располагаются по одному и тому же адресу. Такое наложение полей друг на друга необходимо в тех случаях, когда в разные моменты времени данные, расположенные по этому адресу должны обрабатываться по-разному. Для объявления объединений используется слово union (вместо struct). Длина объединения равна наибольшей длине его полей. В каждый момент времени переменная хранит только одно значение, а ответственность за его правильное использование несет программист. Использование объединений позволяет экономить память в тех случаях, когда известно, что больше одного поля одновременно использоваться не будет. Например, студенты по определенной дисциплине сдают в семестре либо зачет, либо экзамен, поэтому можно не отводить сразу два поля для хранения информации о сданном зачете/экзамене. Определение объединения для этого случая имеет следующий вид:

union prow

{

char sach;

int ekz;

} X;

В случае зачета поле sach будет хранить, например, символ ‘s’ (сдано) или ‘n’(не сдано), а в случае сдачи экзамена поле ekz будет хранить полученную студентом оценку (2, 3, 4 или 5).

Поскольку одновременно по дисциплине сдается или экзамен, или зачет, то в переменной a хранится только одно значение (или символ, или целое число), поэтому для его хранения можно отвести одну и ту же область памяти. Эта область памяти должна быть достаточной для сохранения самого длинного из возможных данных. Переменной X можно присвоить значение любого из двух типов, а затем использовать его в выражениях, но строго по правилам работы с конкретным типом данных. Переменная будет хранить данное того типа, который был присвоен при последнем обращении к переменной. Программист должен следить в своей программе за тем, какое значение было присвоено переменной. Если ей было присвоено значение одного типа, а обработка ведется по правилам, присущим другому типу, то результат будет системно зависимым и трудно предсказуемым. Объединения могут использоваться в структурах и массивах и наоборот. Способ обращения к члену объединения и структуре (или к члену структуры в объединении) полностью совпадает с обращением к элементу вложенной структуры.

**Практика**

1. Создать структуру point с полями-координатами точки. Вычислить расстояние между двумя заданными точками **(lab11\_1.cpp).**
2. Инициализация структур по умолчанию **(lab11\_2.cpp).**
3. Структура в структуре **(lab11\_3.cpp).**
4. Создать массив структур, содержащих два поля – фамилию и рост, упорядочить по полю рост **(lab11\_4.cpp).**
5. Использование структуры для передачи массивов в функцию. В качестве параметра в функцию передается адрес структуры (формальный параметр - простая переменная - указатель).

Необходимо составить функцию, которая на основе матрицы A(mm, nn), где mm≤15, nn≤20, формирует матрицу B по следующему правилу: если элемент матрицы А меньше нуля, то одноименный элемент результирующей матрицы вычисляется как сумма этого элемента и первого элемента строки матрицы А, в которой он расположен, в противном случае очередной элемент определяется как частное от деления элемента на первый элемент строки (предполагается, что первые элементы строк не равны нулю) **(lab11\_5.cpp).**

1. Задание. Создать структуру sport с полями: фамилия спортсмена, название страны, количество оценок спортсмена, массив значений оценок (не более пяти), средняя оценка. Значение поля средняя оценка вычислить при вводе данных. Упорядочить массив по убыванию средней оценки. Определить страну, чьи спортсмены имеют наибольшую сумму средних оценок.
2. **Использование объединений.** Создать массив структур с информацией о фамилии студента, типе контроля знаний (зачет или экзамен) и сданном зачете (экзамене). Упорядочить массив в алфавитном порядке фамилий (**lab11\_6.cpp).**
3. Формирование случайных чисел. Метод Неймана середины квадратов работает так: текущее случайное число возводится в квадрат, после чего выбираются 10 цифр в середине числа. **(lab11\_7.cpp).**
4. Задание. Рассмотреть пример формирования случайных чисел (lab11\_7.cpp). При оценке дублирования чисел использовать механизм создания **исключений.**
5. Задание. Использовать объединения для задания параметров трех разных геометрических фигур: круга, прямоугольника и треугольника. Создать массив структур с информацией о типе геометрической фигуры (круг, прямоугольник, треугольник), ее параметрах (радиус; основание и высота; три стороны). Вычислить значение поля площадь для структур массива.