Лекция 13

Структуры, перечисления: их синтаксис и назначение

Структуры

Структура – это множество поименованных элементов в общем случае разных типов.

Объявление типа **struct** имеет вид:

struct имя_типа {описание элементов};

```
struct book
{
        char title[81];
        int    year;
        int    page;
        float price;
};
```

Элементы структуры называются полями, могут иметь любой тип, в том числе быть указателями на тип самой структуры.

Поля структуры

- Имена полей в структуре должны различаться.
- Имена элементов разных структур могут совпадать.
- Элементом структуры может быть другая структура.

```
struct pets { char name[10];
              int age };
struct boy { char name[10];
             int age;
             pets pet };
```

В языке C++

Тип struct является видом класса и обладает всеми свойствами классов. Все поля класса struct по умолчанию открыты (public), но их можно переопределить как закрытые (private).

Объявление переменных

Следующий оператор:

struct book library;

создает объект типа **struct book**, под который выделяется 110 байт памяти в соответствии со структурным шаблоном.

Используя тип **struct book**, можно описать несколько объектов:

struct book library, catalog[10], *plibrary;

Объявление переменных

Можно объединить описание структурного шаблона и фактическое определение структурной переменной:

```
struct book
{       char title[81];
        int    year;
        int    page;
        float price;
} library;
```

Инициализация и доступ

Структурную переменную можно инициализировать в операторе описания подобно массиву:

struct book library = {“Язык C++”, “Страуструп”, 1990, 500, 1000 };

Доступ к элементам структуры осуществляется с помощью операции точка:

library.author=”Павловская”; // явная инициализация

gets (library.author); //ввод значения

Массивы структур

Описание массива структур аналогично описанию любого другого массива:

struct book catalog[10];

Каждый элемент массива catalog представляет собой структуру типа book.

Для доступа к элементу массива используется индекс, который присоединяется к имени массива:

catalog[2].title

catalog[4].price

catalog[2].title[5] //6 элемент символьного массива в 3-й структуре

Списки структур

Элемент структуры может быть объявлен как указатель на тип структуры, в которую он входит. Это позволяет создавать связанные списки структур:

```
struct sample
{     char    c;
      float   f;
      struct sample *next;    } x;
```

Присвоим переменной x.next адрес динамически выделенной области памяти для хранения значения структуры того же типа:

```
x.next = (struct sample*) malloc (sizeof (x));
```

Вложенные структуры

Элементом структуры может быть другая структура.

```
struct myfile {  char  name[10];
                 char  ftype[4];
                 int  ver; };
struct dir { struct  myfile  f;
             int   size; } my_f [100];
```

Шаблон для вложенной структуры должен располагаться перед определением фактической структурной переменной в рамках другой структуры.

```
my_f [0].size    //элемент size 1-ой структуры
my_f [2].f.ver   //элемент ver вложенной структуры f в 3-й структуре my_f
```

Указатели на структуры

Объявим указатель на структуру:

struct dir *pst;

Указатель *pst можно использовать для ссылок на любые структуры типа dir:

pst = &my_f [0];

Указатели на структуры

Доступ к полям структуры через указатель осуществляется с помощью операции косвенного доступа

pst->size ≡ my_f [0].size

(pst+i)->size ≡ my_f [i].size

Для вложенных структур:

my_f [0].f.ver ≡ pst (f.ver)

my_f [0].f.name[0] ≡ pst (f.name[0])

Структуры и функции

Для передачи информации о структуре внутрь функции используются следующие способы:

- Использование в качестве фактического аргумента элемента структуры.
- Использование в качестве фактического аргумента адреса структуры.
- Использование в качестве фактического аргумента самой структуры.

Перечисления в C++

Перечисление - это тип данных, который описывает набор именованных целочисленных констант.

Переменным перечислимого типа можно присваивать только именованные значения перечислимых констант.

Целочисленным переменным можно присваивать значения перечислимых констант.

Применение перечислений делает программы нагляднее.

Объявление перечислимого типа:

```
enum  color {red, green, black}; // вариант 1
 const int red = 0;
 const int green = 1;
 const int black = 2;

 enum  color
     { red = 2, green = 2, black = 6 }; // вариант 2
```

Объявление переменной перечислимого типа:

```
enum  color c;
        color d;
enum  color  {red, green, black} a;
```

```
enum  color c = red;
int  i = red;
```

Пример. Использование перечислимого типа

```
#include <iostream>
using namespace std;
#include <conio.h>

enum itsaWord { NO, YES };
int main()
{ itsaWord isWord = NO;
  char ch = 'a';
  int wordcount = 0;
  cout << "Enter a phrase:\n";
  do
  { ch = getche();
    if(ch ==' ' || ch =='\r')
     { if( isWord == YES )
        { wordcount++;
         isWord = NO;
         }
     }
    else if( isWord == NO )
             isWord = YES;
 } while( ch != '\r' );   //quit on Enter key
   cout << "\n---Word count is "
        << wordcount << "---\n";
   return 0;
}
```

Программа подсчитывает количество слов в предложении, вводимом с клавиатуры. Слова разделяются одним или несколькими пробелами.

```
Enter a phrase:
abcd  dfe gfe
---Word count is 3---
```