

Структуры и объединения



Структуры

- **Структура** – это множество поименованных элементов в общем случае разных типов.
- Объявление типа ***struct*** имеет вид:
struct имя_типа {описание элементов};
- Элементы структуры называются полями, могут иметь любой тип, в том числе быть указателями на тип самой структуры.

```
struct book
{
    char title[81];
    char author[21];
    int year;
    int page;
    float price;
};
```



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 };
```

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;
```

Можно объединить описание структурного шаблона и фактическое определение структурной переменной:

```
struct book  
{  
    char title[81];  
    char author[21];  
    int year;  
    int page;  
    float price;  
} library;
```

Инициализация и доступ

- Элементы структуры в памяти запоминаются последовательно, в том порядке, в котором они объявляются: первому элементу соответствует меньший адрес памяти, последнему - больший.
- Структурную переменную можно инициализировать в операторе описания подобно массиву:

```
struct book library = {    "Язык С++",  
                        "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
```

Массивы структур

- Вывести фамилию студента, получающих самую высокую стипендию и фамилии студентов, начинающихся на букву «А»
- `#include "stdafx.h"`
- `#include<iostream>`
- `using namespace std;`
- `struct student {`
- `char name[15];`
- `int kurs;`
- `int step;`
- `};`
- `int main()`
- `{ int i, max, maxi;`
- `student st[3];`
- `for (i = 0; i < 3; i++)`
- `{ cout << "name: "; cin >> st[i].name;`
- `cout << "kurs: "; cin >> st[i].kurs;`
- `cout << "step: "; cin >> st[i].step;`
- `}`
- `}`
-

Массивы структур

- `max = 0; maxi = 0;`
- `for (i = 0; i < 3; i++)`
- `if (max < st[i].step) { max = st[i].step; maxi = i; };`
- `cout << "max stependiya " << st[maxi].name<<endl;`
- `for (i = 0; i < 3; i++)`
- `{`
- `if (st[i].name[0] == 'A')`
- `cout << st[i].name << endl;`
- `}`
- `for (i = 0; i < 3; i++)`
- `{`
- `cout << "name: " << st[i].name;`
- `cout << "kurs: " << st[i].kurs;`
- `cout << "step: " << st[i].step<<endl;`
- `}`
- `return 0;`
- `}`

Массивы структур

```
C:\Windows\system32\cmd.exe
name: Aleksei
kurs: 2
step: 123
name: Bota
kurs: 3
step: 102
name: Artem
kurs: 3
step: 132
max stependiya Artem
Aleksei
Artem
name: Alekseikurs: 2step: 123
name: Botakurs: 3step: 102
name: Artemkurs: 3step: 132
Для продолжения нажмите любую клавишу . . .
```

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];`
- 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.

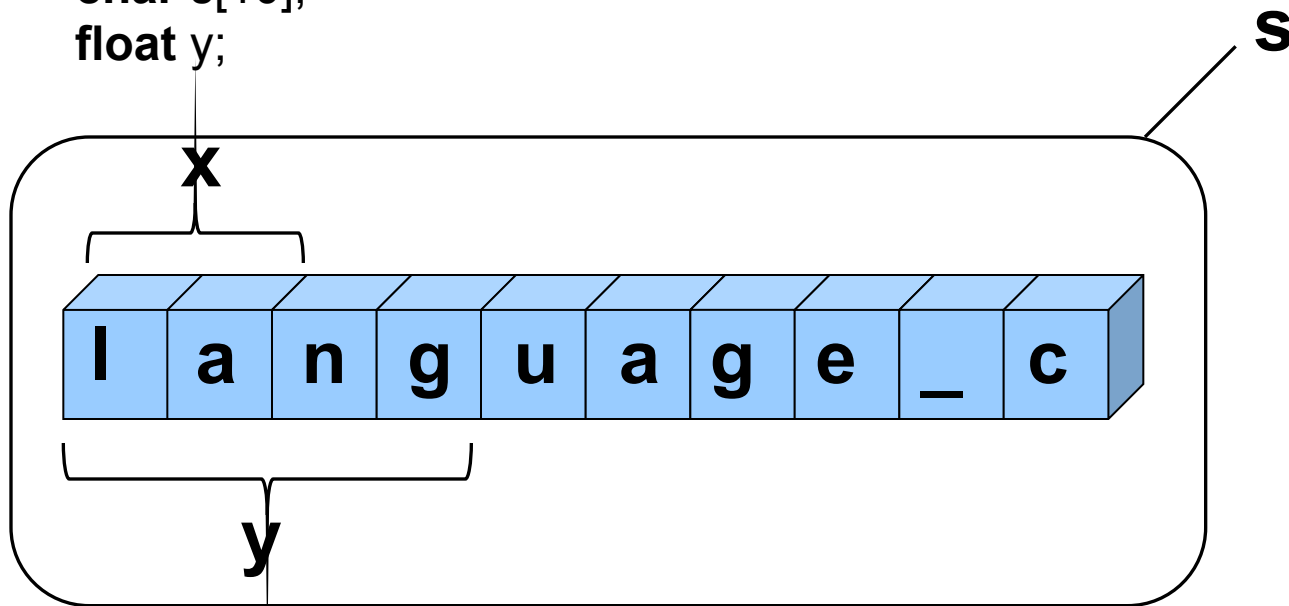
Объединения

- Объявление объединения синтаксически совпадает с объявлением структуры, но начинается с ключевого слова ***union***.
- Поля объединения хранятся в одной области памяти и имеют один и тот же начальный адрес.
- Размер объединения определяется максимальным размером его элементов.
- Объединения экономят память и используются, если в каждый момент времени используется только одно поле.

```
union small {  
    int x;  
    char s[10];  
    float y;  
} val;
```
- Под объект `val` память выделяется в соответствии с размером максимального поля - 10 байт.

Пример (стандарт IEEE-754)

```
union small {int x;  
             char s[10];  
             float y;  
} val;
```



val.s[] = "language_c"

x = 24940 = 0x6C61

y = 1.1257203e+24 = 0x6C616E67

'l' = 0x6C

'a' = 0x61

'n' = 0x6E

'g' = 0x67

Перечислимые типы данных

- Перечислимый тип (перечисление) – определение целочисленных констант, для каждой из которых вводятся имя и значение.
- Объявление:
`enum имя_типа {список перечисления} идентификатор;`
- Пример:
`enum progr {C, Pascal, Foxpro, Modula 2, Basic, Fortran} lang;`
- Пример явной инициализации констант:
`enum mas {elem1 = -1, elem2, elem3 = 5};`
- Элементам перечисления могут быть присвоены одинаковые значения:
`enum mas {elem1 = 2, elem2, elem3 = 2, elem4};`