

Структуры и объединения

лекция 7

План лекции

- Понятие структуры
- Описание структур и переменных типа структура
 - Незавершённые типы данных
- Операции над значениями типа структура
 - Операция присваивания структур
- Инициализация переменных типа структура
- Размещение структур в памяти
- Объединения

Понятие структуры

- "Структура — это объект, состоящий из последовательности именованных элементов различных типов." (K & R, 3е издание)
- Элементы, из которых состоит структура, также называются *полями* и/или *членами*
- С точки зрения математики, структуры – это функции, заданные в конечном количестве "точек"
 - Значение типа структура отображает явно заданное конечное множество идентификаторов (полей) в значения явно заданных типов (типов полей) так, что каждый идентификатор всегда отображается в значения своего фиксированного типа
- Структуры – это ещё один вид составных типов данных в языке Си

Описание структур и переменных типа структура

- Описание типа данных "структура S" с полями E1, ..., EN типов T1, ..., TN имеет вид
struct S { T1 E1; ... TN EN; };
- Описание типа данных "структура S" с полями E1, ..., EN типов T1, ..., TN можно совместить с описанием переменных типа "структура S"
struct S { T1 E1; ... TN EN; } V, W, ...;
- Если идентификатор после ключевого слова struct пропущен, то такой тип данных называется *анонимная структура*
struct { T1 E1; ... TN EN; } V, W, ...;
- Идентификаторы полей в структурах разных типов могут совпадать

Описание структур, объявление переменных типа структура

```
struct my_point {  
    int x;  
    int y;  
};
```

```
struct my_rect {  
    struct my_point top_left;  
    struct my_point bottom_right;  
};
```

```
struct my_point p, q;  
struct my_rect rect1, rect2;
```

Описание структур, объявление переменных типа структура

```
struct my_point {  
    int x, y;  
};
```

```
struct my_rect {  
    struct my_point top_left, bottom_right;  
};
```

Незавершённые типы

- *Предписание* типа данных "структура S" имеет вид
struct S;
- Тип данных может иметь любое число предписаний и только одно описание
- После предписания тип данных является *незавершённым* до тех пор, пока компилятор не встретит его описание
- Указатель на незавершённый тип является обычным типом (не незавершённым)
- Любое другое использование незавершённого типа вызывает ошибку компиляции

Незавершённые типы

```
struct my_int_list {  
    int value;  
    // struct my_int_list – незавершённый тип  
    // struct my_int_list * -- указат. на незав. тип  
    struct my_int_list *next_element;  
};
```

```
struct my_incomplete_int_list {  
    int value;  
    // ошибка компиляции, т.к.  
    // незавершённый тип  
    // struct my_incomplete_int_list нельзя  
    // использовать для описания поля структуры  
    struct my_incomplete_int_list next_element;  
};
```

Операции над значениями типа структура

- `my_struct_var.my_field`
 - Результат – значение поля `my_field` переменной `my_struct_var` типа структура
 - В описании типа переменной `my_struct_var` должно встречаться поле `my_field`
- `ptr_to_my_struct_var->my_field`
 - Сокращение для `(*ptr_to_my_struct_var).my_field`
- `&my_struct_var`
 - Результат – адрес значения переменной `my_struct_var` типа структура

Операция присваивания для структур

- `my_struct_var1 = my_struct_var2`
 - Результат -- `my_struct_var2`, если переменные `my_struct_var1` и `my_struct_var2` имеют одинаковый тип
- Если идентификаторы `S1` и `S2` различны, то типы "структура `S1`" и "структура `S2`" различны
- Каждый тип "анонимная структура" отличается от всех других типов

Операция присваивания для структур

```
struct {int x,y;} p, q;  
p = q; // ОК
```

```
struct my_pt {int x, y;} p;  
struct my_pt q;  
p = q; // ОК
```

```
struct {int x,y;} p;  
struct {int x,y;} q;  
p = q; // ошибка, т.к. p и q имеют разные  
ТИПЫ
```

Операция присваивания для структур

```
void f(struct {int x,y;} p) { /* ... */ }
```

```
int main(void)
{
    struct {int x,y;} q;
    f(q); // ошибка, т.к. факт. и форм. парам.
        // разных типов
    return 0;
}
```

Операция присваивания для структур

```
struct my_pt {int x,y};
```

```
void f(struct my_pt p) { /* ... */ }
```

```
int main(void)
{
    struct my_pt q;
    f(q); // OK
    return 0;
}
```

Инициализация структур

- Описание переменных типа структура может задавать начальные значения полей структуры
- C89

```
struct S { T1 E1; ... TN EN; } V = { И1, ..., ИК };
```

 - если $K > N$, то ошибка компиляции
 - И1 – инициализатор поля E1 и т.д.
 - Память, отведённая под значения полей после поля EK, заполняется байтом 0
- C99 и C11

```
struct S { T1 E1; ... TN EN; } V =  
    { .E1 = И1, ..., .EK = ИК };
```

Инициализация структур

```
struct my_point {  
    int x;  
    int y;  
};
```

```
struct my_rect {  
    struct my_point top_left;  
    struct my_point bottom_right;  
};
```

```
struct my_point p = {0, 0}, q = {100, 100};  
struct my_rect rect1 = {p, q};
```

```
// Вложенные инициализаторы  
struct my_rect rect2 = {{0,0}, {100, 100}};
```

Размещение структур в памяти

- Адреса полей структуры возрастают по мере объявления
- Адрес структуры равен адресу первого поля структуры
- Размер структуры \geq суммы размеров её полей
 - Компилятору разрешено оставлять пустое место между полями структуры, чтобы обеспечить выравнивание значений простых типов по правилам Си (см. Лекцию 3, слайд 21)

Объединения

- "Объединение — объект, который в каждый момент времени хранит значение одного из нескольких различных типов." (K&R, 3е издание)
- Для описания объединений используется ключевое слово `union`
- "Объединение можно представить себе как структуру, все элементы которой начинаются со смещением 0 и размеры которой достаточны для хранения любого из элементов." (K&R, 3е издание)
- Объединения – это последний вид составных типов данных в языке Си

Размещение объединений в памяти

- Адреса всех полей объединения совпадают с адресом объединения

```
union {long long LL; double D;} my_union_var;  
// &my_union_var == &my_union_var.LL  
// &my_union_var == &my_union_var.D
```
- Если поля объединения – структуры, то для размещения полей этих структур действуют общие правила размещения структур в памяти

Размещение объединений в памяти

```
struct var_t {char name[16]; int init_value;};
```

```
struct num_t {int value;};
```

```
struct expr_t; // преодобъявление
```

```
struct bin_op_t {  
    enum {ADD, SUB, MUL, DIV} op;  
    struct expr_t *left, *right;  
};
```

```
struct expr_t {  
    enum {VAR, NUM, BIN_OP} type;  
    union {var_t var; num_t num; bin_op_t op;} body;  
};
```

Размещение объединений в памяти

```
struct var_t  create_var(const char name[], int init_val)
{
    struct var_t v = {name, init_val};
    return v;
}
struct var_t  create_num(int value)
{
    struct var_t n = {value};
    return n;
}
struct bin_op_t  create_bin_op(enum op_t op, const struct expr_t *L, const struct
expr_t *R)
{
    struct bin_op_t o = {op, L, R};
    return o;
}
```

Заключение

- Понятие структуры
- Описание структур и переменных типа структура
 - Неаворщенные типы данных
- Операции над значениями типа структура
 - Операция присваивания структур
- Инициализация переменных типа структура
- Размещение структур в памяти
- Объединения