

Структури в C/C++

Структура в мові C(C++) – це агрегатний тип даних, який складається з визначеної кількості елементів різних типів, що називаються **членами структури** (інколи – **полями структури**). Синтаксис визначення структури:

```
struct tag_name
{
    <тип_1> mem_1; //перший член структури
    <тип_2> mem_2; //другий член структури
    ...
    <тип_n> mem_n; //останній член структури
}; // крапка з комою обов'язкова
```

Визначення екземпляру структури:

```
struct tag_name id_example; // стиль C
tag_name id_example; // стиль C++
```

Доступ до членів структури – через крапкову нотацію.

Зауваження

1. При визначенні структури тег не є обов'язковим. Але тоді екземпляри структури, визначеної анонімно – без тегу, мають визначатись одразу після фігурної дужки, що закриває тіло структури.
2. При визначенні екземпляру структури в мові C вживання службового слова **struct** обов'язкове, а в C++ – може бути пропущене.
3. Обмежень на типи членів структури немає, крім одного – членом структури не може бути екземпляр даної структури, проте може бути вказівник на неї (Це дозволяє створювати динамічні структури даних – списки, дерева, тощо).
4. Структури можуть бути параметрами функцій та повертатись як результат функції.
5. В C++ членами структури можуть бути функції (методи), зокрема - конструктор.
6. Вважається хорошим стилем програмування тег структури записувати великими літерами – так структури краще відрізняти від звичайних змінних.

Приклад.

```
// Визначаємо полярні координати
struct POLAR
{
    float r;
    float phi;
};
// Визначаємо полярні точки a, b
struct POLAR a, b;
// Доступ до членів структури:
    a.r    = 1;
    a.phi  = 0;
// Структуру можна ініціалізувати
struct POLAR c = {1, M_PI*0.5};
```

Допустимі операції зі структурами:

- доступ до членів структури;
- копіювання й присвоєння структур;
- взяття адреси структури.

Структури також можуть передаватись у функції в ролі параметрів (за значенням) і повертатись як результат функції.

Приклад (продовження)

```
struct POLAR
{
    float r, phi;
};
struct POLAR a, b, *p;
a.r    = 1;
a.phi  = 0;
b = a;    // копіювання структури
p = &b;   // взяття адреси структури
(*p).r = 5; // дужки обов'язкові - визначають порядок операцій
```

Для спрощення запису доступу до членів структури, що адресується вказівником, використовується спеціальна операція, яка позначається `->` (знак `-` і знак `>`). Отже, останній рядок можна записати таким чином:

```
p -> r = 0;
```

Зверніть увагу, ліворуч від `->` знаходиться вказівник на структуру, праворуч – член структури.

Переліки в C/C++

Перелік в мові C(C++) – це тип даних, який використовується для створення набору іменованих констант. Синтаксис визначення переліку:

```
enum <tag_name>
{
<const_1> [=<value_1>],
<const_2> [=<value_2>],
...
<const_n> [=<value_n>],
};
```

Якщо значення констант вказані не всі, або не вказані взагалі, то присвоюються значення з кроком +1, починаючи від останньої, або від 0, починаючи з першої.

Приклад.

```
/* Використовуємо перелік для назв днів
тижня */
enum Days
{
    Mn = 1, Tu, Wn, Th, Fr, Sa, Su
};
int main()
{
    enum Days d;
    d = Th;
    cout << d; // виведеться значення 4
    system ("PAUSE");
    return 0;
}
```

Об'єднання в C/C++

Об'єднання в мові C(C++) – це тип даних, який складається з визначеної кількості елементів, що називаються **членами об'єднання** (інколи – **полями об'єднання**). В пам'яті знаходиться завжди один і тільки один член об'єднання. Тобто об'єднання – це структура із нульовим зміщенням кожного поля відносно її початку. Синтаксис об'єднання :

```
union tag_name
{
    <тип_1> mem_1; //перший член об'єднання
    <тип_2> mem_2; //другий член об'єднання
    ...
    <тип_n> mem_n; //останній член об'єднання
}; // крапка з комою обов'язкова
```

Визначення екземпляру об'єднання :

```
union tag_name id_example; // стиль C
tag_name id_example; // стиль C++
```

Доступ до членів об'єднання – через крапкову нотацію або операцію \rightarrow для вказівників, так само як і для структури.

Зауваження

1. Всі зауваження, які були зроблені щодо структур, справедливі і для об'єднань.
2. Об'єднання часто використовуються для неявного приведення типів – наприклад, можна записати одним членом об'єднання дійсне значення, а прочитати з іншого – ціле. Проте такі дії можуть привести до неочікуваних побічних ефектів.

Приклад.

```
/* Використовуємо об'єднання для приведення типів -
   результат сумнівний */
union casting
{
    int i;
    float f;
    unsigned u;
};
int main()
{
    union casting cast = {-1}; /* ініціалізувати
    можна лише перший член об'єднання */
    cout << "\nfloat member: " << cast.f << endl;
    cout << "\nint member: " << cast.i << endl;
    cout << "\nunsigned member: " << hex << cast.u
        << endl;
    system ("PAUSE");
return 0;
}
```

Бітові поля в C/C++

Бітові поля в мові C(C++) – це структура, яка складається з визначеної кількості окремих бітів вказаної довжини. Синтаксис структури – бітове поле:

```
struct bit_field
{
<тип_1> mem_1 : n1; // n1 - довжина в бітах
<тип_2> mem_2 : n2; // n2 - довжина в бітах
    ...
<тип_n> mem_n : nn; // nn - довжина в бітах
};
```

Можна використати об'єднання, членами якого є ціле число та бітове поле – це забезпечить простий доступ до окремих бітів цілого числа!