Структуры

Определение структур

Последнее обновление: 01.06.2017

Структура в языке программирования Си представляет собой производный тип данных, который объединяет в единое целое множество компонентов. При этом в отличие от массива эти компоненты могут представлять различные типы данных.

Для определения структуры применяется ключевое слово **struct**, а сам формат определения выглядит следующим образом:

```
struct имя_структуры
{
    компоненты_структуры
};
```

Имя_структуры представляет произвольный идентификатор, к которому применяются те же правила, что и при наименовании переменных.

После имени структуры в фигурных скобках помещаются *Компоненты_структуры*, которые представляют набор описаний объектов, которые составляют структуру.

Следует отметить, что в отличие от функции при определении структуры после закрывающей фигурной скобки идет точка с запятой.

Например, определим простейшую структуру:

```
struct person
{
    int age;
    char * name;
};
```

Здесь определена структура person, которая имеет два элемента: age (представляет тип int) и name (представляет указатель на тип char).

Все элементы структуры объявляются как обычные переменные. Но в отличие от переменных при определении элементов структуры для них не выделяется память, и их нельзя инициализировать. По сути мы просто

определяем новый тип данных.

После определения структуры мы можем ее использовать. Для начала мы можем определить объект структуры - по сути обычную переменную, которая будет представлять выше созданный тип:

```
struct person tom;
```

Здесь определена переменная tom, которая представляет структуру person. И при каждом определении переменной типа структуры ей будет выделяться память, необходимая для хранения ее элементов.

При определении переменной структуры ее можно сразу инициализировать, присвоив какое-нибудь значение:

```
struct person tom = {23, "Tom"};
```

Инициализация структур аналогична инициализации массивов: в фигурных скобках передаются значения для элементов структуры по порядку. Так как в структуре person первым определено свойство, которое представляет тип int - число, то в фигурных скобках вначале идет число. И так далее для всех элементов структуры по порядку.

Также после создания переменной структуры можно обращаться к ее элементам - получать их значения или, наоборот, присваивать им новые значения. Для обращения к элементам структуры используется операция "точка":

```
имя_переменной_структуры.имя_элемента
```

Теперь объединим все вместе в рамках программы:

```
#include <stdio.h>

struct person
{
    int age;
    char * name;
};

int main(void)
{
    struct person tom = {23, "Tom"};
    printf("Age: %d \t Name: %s", tom.age, tom.name);
    return 0;
}
```

Консольный вывод программы:

```
Age: 23    Name: Tom
```

С элементами структуры можно производить все те же операции, что и с переменными тех же типов. Например, добавим ввод с консоли:

```
#include <stdio.h>

struct person
{
    int age;
    char name[20];
};
int main(void)
{
    struct person tom = {23, "Tom"};
    printf("Enter name: ");
    scanf("%s", &tom.name);
```

```
    printf("Enter age: ");
    scanf("%d", &tom.age);
    printf("Name:%s \t Age: %d", tom.name, tom.age);
    return 0;
}
```

Консольный вывод программы:

```
Enter name: Eugene
Enter age: 33
Name: Eugene        Age: 33
```

Мы можем одновременно совмещать определение типа структуры и ее переменных:

```
#include <stdio.h>

struct person
{
    int age;
    char * name;
} tom, bob, alice;

int main(void)
{
    tom.name ="Tom";
    tom.age = 23;
    bob.name ="Bob";
    bob.age = 15;
    alice.name = "Alice";
    alice.age=25;
    printf("Name:%s \t Age: %d", tom.name, tom.age);
    return 0;
}
```

После определения структуры, но до точки запятой мы можем через запятую перечислить набор переменных. А затем присвоить их элементам значения.

При подобном определении мы можем даже не указывать имя структуры:

```
struct
{
    int age;
    char * name;
} tom, bob, alice;
```

В этом случае компилятор все равно будет знать, что переменные tom, bob и alice представляют структуры с двумя элементами name и age. И соответственно мы также с этими переменными сможем работать. Другое дело, что мы не сможем задать новые переменные этой структуры в других местах программы.

typedef

Еще один способ определения структуры представляет ключевое слово **typedef**:

```
#include <stdio.h>

typedef struct
{
    int age;
    char * name;
} person;
```

```
int main(void)
{
    person tom = {23, "Tom"};
    printf("Name:%s \t Age: %d", tom.name, tom.age);
    return 0;
}
```

В конце определения структуры после закрывающей фигурной скобки идет ее обозначение - в данном случае person. В дальнейшем мы можем использовать это обозначение для создания переменной структуры. При этом в отличие от примеров выше здесь не надо при определении переменной не надо использовать слово **struct.**

Директива define

Еще один способ определить структуру представляет применение препроцессорной директивы **#define:**

```
#include <stdio.h>

#define PERSON struct {int age; char name[20];}

int main(void)
{
    PERSON tom = {23, "Tom"};
    printf("Name:%s \t Age: %d", tom.name, tom.age);
    return 0;
}
```

В данном случае директива define определяет константу PERSON, вместо которой при обработке исходного кода препроцессором будет вставляться код структуры struct {int age; char name[20];}

Назад Содержание Вперед

ALSO ON **METANIT.COM**

Уязвимости и языки программирования ...

16 дней назад • 1 comment

Уязвимости и языки программирования, сторонние библиотеки, …

Множественная регистрация ...

2 месяца назад • 2 comments

Множественная регистрация сервисов в ASP.NET Core и C#, …

Конфигурация и генерация ...

3 месяца назад • 3 comments

Конфигурация и генерация JWT-токенов для аутентификации и …

Авториз помощь

3 месяца н

Авторизац JWT-токе приложен

Помощь сайту
YooMoney

- 410011174743222

Перевод на карту

- Номер карты: 4048415020898850
- Номер карты: 4890494751804113
