

# Программирование на C++



Минцифры  
России

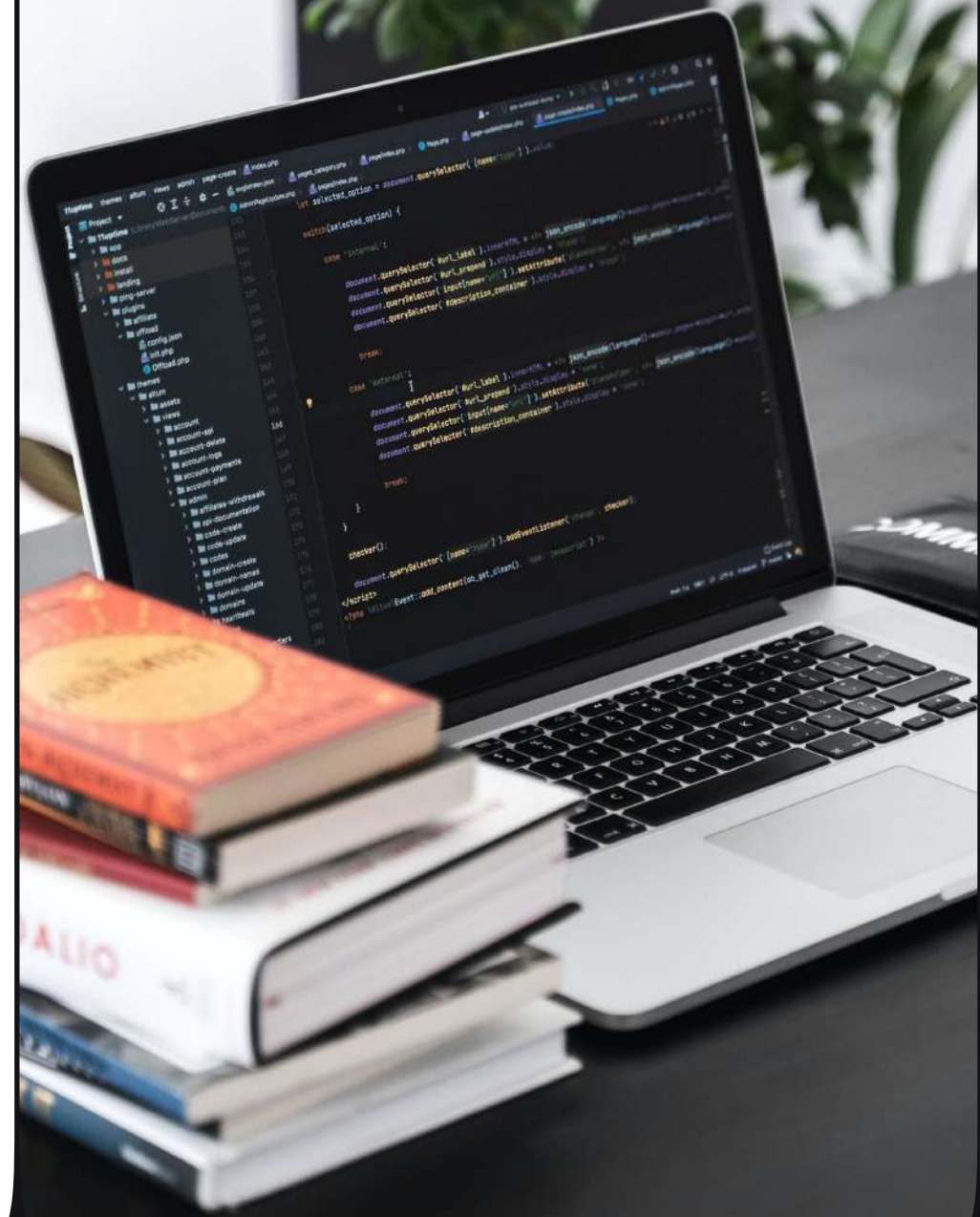
UCHi **DOMA**

**20.35**  
УНИВЕРСИТЕТ

Урок 8 Модуль 2

# Основные сведения о структурах

Полезные материалы





# Цели урока

- ✦ изучить хранение данных в структурах
- ✦ отработать на практике составление алгоритмов с хранением данных в структурах на Си



# Структура

**Структура** — это объединение нескольких объектов, возможно, различного типа под одним именем, которое является типом структуры. В качестве объектов могут выступать переменные, массивы, указатели и другие структуры.



Структуры позволяют трактовать группу связанных между собой объектов не как множество отдельных элементов, а как единое целое.



Структура представляет собой сложный тип данных, составленный из простых типов.

# Объявление структур

Общая форма объявления структуры:

```
struct ИмяСтруктуры  
{  
    тип ИмяЭлемента1;  
    тип ИмяЭлемента2;  
    . . .  
    тип ИмяЭлементаn;  
};
```



**Внимание!** После закрывающей фигурной скобки } в объявлении структуры обязательно ставится точка с запятой.

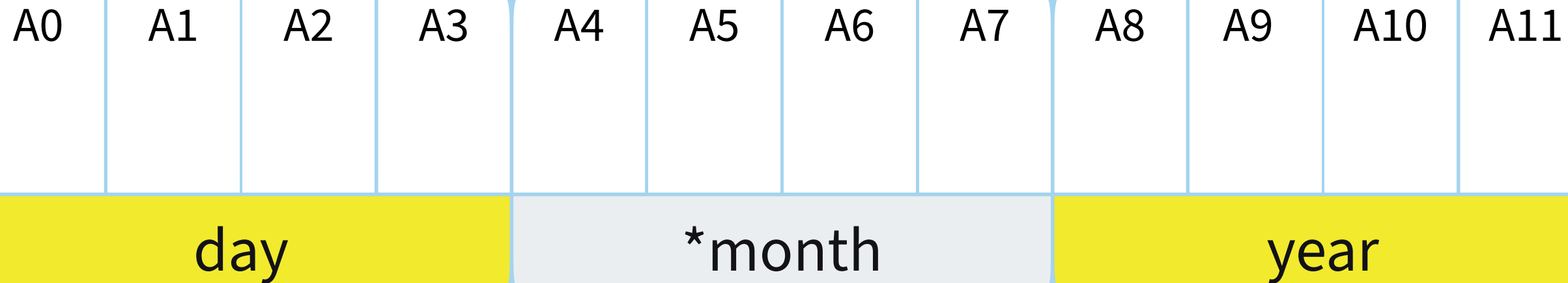


# Пример



```
struct date
{
    int day; // 4 байта
    char *month; // 4 байта
    int year; // 4 байта
};
```

Поля структуры располагаются в памяти в том порядке, в котором они объявлены:



В указанном примере структура date занимает в памяти 12 байт. При объявлении структур, их разрешается вкладывать одну в другую.

# Пример



```
struct persone
{
char lastname[20]; // фамилия
char firstname[20]; // имя
struct date bd; // дата рождения
};
```

# Инициализация полей структуры

Инициализация полей структуры может осуществляться двумя способами:

- 1 присвоение значений элементам структуры в процессе объявления переменной, относящейся к типу структуры
- 2 присвоение начальных значений элементам структуры с использованием функций ввода-вывода (например, `printf()` и `scanf()`)



# Первый способ

В первом способе инициализация осуществляется по следующей форме:

```
struct ИмяСтруктуры ИмяПеременной={ЗначениеЭлемента1, ЗначениеЭлемента_2,..., ЗначениеЭлементаn};
```

**Пример:**

```
struct date bd={8,"июня", 1978};
```

# Обращение к структуре

Имя элемента структуры является составным. Для обращения к элементу структуры нужно указать имя структуры и имя самого элемента. Они разделяются точкой:

ИмяСтруктуры.ИмяЭлементаСтруктуры

```
printf("%d %s %d",bd.day, bd.month, bd.year);
```

# Второй способ

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <stdlib.h>
4  struct date {
5      int day;
6      char month[20];
7      int year;
8  };
9  struct persone {
10     char firstname[20];
11     char lastname[20];
12     struct date bd;
13 };
14 int main() {
15     system("chcp 1251");
16     system("cls");
17     struct persone p;
18     printf("Введите имя : ");
19     scanf("%s", p.firstname);
20     printf("Введите фамилию : ");
21     scanf("%s", p.lastname);
22     printf("Введите дату рождения\nЧисло: ");
23     scanf("%d", &p.bd.day);
24     printf("Месяц: ");
25     scanf("%s", p.bd.month);
26     printf("Год: ");
27     scanf("%d", &p.bd.year);
28     printf("\nВы ввели : %s %s, дата рождения %d %s %d года",
29         p.firstname, p.lastname, p.bd.day, p.bd.month, p.bd.year);
30     getchar(); getchar();
31     return 0;
}
```

Инициализация объектов языка Си  
с использованием функций  
ВВОДА-ВЫВОДА

# Указываем имя при объявлении

Имя структурной переменной может быть указано при объявлении структуры.

В этом случае оно размещается после закрывающей фигурной скобки }. Область видимости такой структурной переменной будет определяться местом описания структуры.

```
1 struct image // имя структуры
2 {
3     double vertical;
4     double horizontal;
5     double color;
6     double size;
7 } pic1; // имя структурной переменной
```

# Объединения



Объединениями называют сложный тип данных, позволяющий размещать в одном и том же месте оперативной памяти данные различных типов.



Размер оперативной памяти, требуемый для хранения объединений, определяется размером памяти, необходимым для размещения данных того типа, который требует максимального количества байт.



Когда используется элемент меньшей длины, чем наиболее длинный элемент объединения, то этот элемент использует только часть отведенной памяти.



Все элементы объединения хранятся в одной и той же области памяти, начиная с одного адреса.



# Объявление объединений

## Общая форма объявления объединения

```
union ИмяОбъединения
{
    тип ИмяОбъекта1;
    тип ИмяОбъекта2;
    . . .
    тип ИмяОбъектаn;
};
```

# Применение объединений

Объединения применяются для следующих целей:



для инициализации объекта, если в каждый момент времени только один из многих объектов является активным



для интерпретации представления одного типа данных в виде другого типа

# Пример



Задача:

вещественное число типа float представить в виде совокупности байтов

```
1  #define _CRT_SECURE_NO_WARNINGS
2  #include <stdio.h>
3  #include <stdlib.h>
4  union types
5  {
6      float f;
7      unsigned char b[4];
8  };
9  int main()
10 {
11     types value;
12     printf("N = ");
13     scanf("%f", &value.f);
14     printf("%f = %x %x %x %x", value.f, value.b[0], value.b[1], value.b[2], value.b[3]);
15     return 0;
16 }
```

# Пример



## Результат работы программы

$N = 15.3$

$15.300000 = cd\ cc\ 74\ 41$