Семинар 5
Структуры, объединения, битовые поля

1. Структуры

Из основных типов языка Си++ пользователь может конструировать производные типы. Наиболее значимым производным типом является класс.

Структуры и объединения – частные случаи класса.

1. Структуры

Простейший формат определения (спецификации) класса:

ключ_класса имя_класса
{компоненты класса};

1. Структуры

ключ_класса – это одно из служебных слов: class, struct, union.

имя_класса – идентификатор, выбираемый программисом

компоненты класса – типизированные данные (поля или свойства) и принадлежащие классу функции (методы).

1. Структуры

Если в классе отсутствуют явные определения методов, а в качестве ключа использовано ключевое слово struct, то класс соответствует *структурному типу*. Если методы отсутствуют и используется ключевое слово union, то создаётся *объединяющий тип*.

1. Структуры

Структура – это объединённое в единое целое множество поименованных элементов в общем случае разных типов.

1. Структуры

Пример. Создание нового типа book.

```
struct book{
    char author[30]; // имя автора
    char title[80]; // название
    char city[20]; // город
    char firm[20]; // издательство
    int year; // год
    int pages; // количество страниц
} b1, b2, bArr[8], *bPtr;
```

1. Структуры

Где:
b1, b2 – структуры типа book,
bArr – массив структур типа book,
bPtr – указатель на структуру.

1. Структуры

Обращение к полям данных:
имя_структуры.имя_элемента
имя_указателя -> имя_элемента
(*имя_указателя) -> имя_элемента

1. Структуры

Пример обращения к полям данных:

```
#define PRINT(c) cout << #c << "=" << c << endl;
book myBook = {"Podbelsky V.V.",
"Standartny C++", "Moscow",
"Finansy i statistika", 2008, 688};
book * myBookPtr = &myBook;
PRINT(myBook.author);
PRINT(myBookPtr->author);
…
```

1. Структуры

Для структур могут быть определены ссылки:

```
book & refBook = myBook;
```

или

```
book & refBook(myBook);
```

1. Структуры

В отношении элементов структур существует одно существенное ограничение – элемент структуры не может иметь тот же самый тип, что и определяемый структурный тип. В то же время элементом определяемой структуры может быть указатель на структуру определяемого типа.

1. Структуры

Пример.

```
// Ошибка:
struct mistake {mistake s; int m;}
// Корректное определение:
struct ok {ok * ptr; int m;}
```

1. Структуры

Функции могут возвращать и использовать в качестве параметров структуры, указатели на структуры, ссылки на структуры.

```
int f1(book b); // Передача по значению
int f2(book * b); // Параметр-указатель
int f3(book & b); // Параметр-ссылка
```

1. Структуры

Операндом для операции new может быть структурный тип. Пример использования:

```
book * bookPtr, booksPtr;
bookPtr = new book;
booksPtr = new book[100];
```

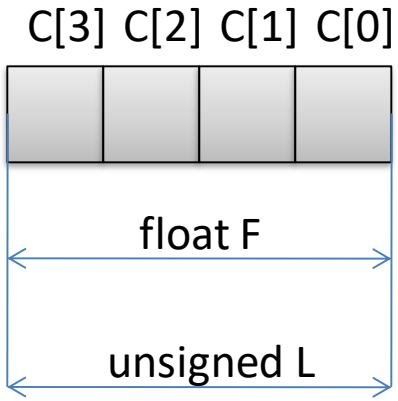
2. Объединения

Основное назначение объединений – обеспечить возможность доступа к одному и тому же участку памяти с помощью объектов различных типов.

2. Объединения

Пример. Объединение для вывода внутренней структуры значений типа double и float:

```
union U4{
    float F;
    unsigned L;
    char C[4];
} FLC;
```



2. Объединения

Пример. Объединение для вывода внутренней структуры значений типа double и float:

```
FLC.L = 10;
cout << "1st byte: " << int(FLC.C[0]) << endl;
cout << "2nd byte: " << int(FLC.C[1]) << endl;
cout << "3rd byte: " << int(FLC.C[2]) << endl;
cout << "4th byte: " << int(FLC.C[3]) << endl;
```

2. Объединения

При определении конкретных объединений разрешена их инициализация, причём инициализируется только первый элемент объединения

```
U4 ok = {10.4};
U4 notOk = {'f', 'a', '0', '4'}; // Ошибка
```

3. Битовые поля

Каждое битовое поле представляет собой целое или беззнаковое целое значение, занимающее в памяти целое число битов (например, 1 бит, 5 бит, 16 бит). Битовые поля могут быть только элементами классов. Назначение битовых полей – обеспечить удобный доступ к отдельным битам данных.

3. Битовые поля

Определение структуры с битовыми полями:

```
struct имя_структурного_типа{
        тип_поля имя_поля:ширина_поля;
        тип_поля имя_поля:ширина_поля;
} имя_структуры;
```

тип_поля – один из базовых целых типов (int, char, short, long)

3. Битовые поля

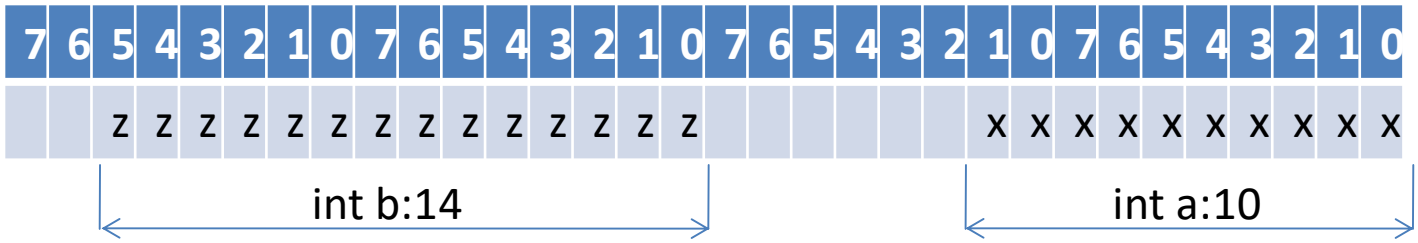
Пример определения структуры с битовыми полями:

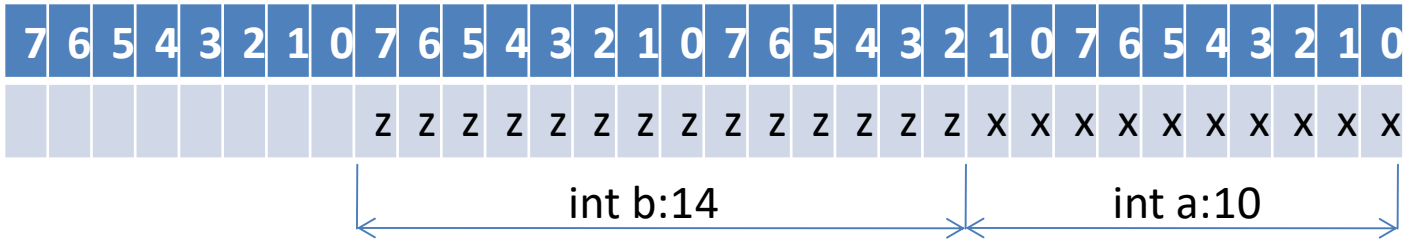
```
struct {
        int a:10;
        int b:14;
} x;
```

3. Битовые поля

```
struct {
        int a:10;
        int b:14;
} x;
```

Порядок размещения полей (справа налево или слева направо) и порядок размещения полей, длина которых не кратна длине слова или длине байта, зависят от реализации

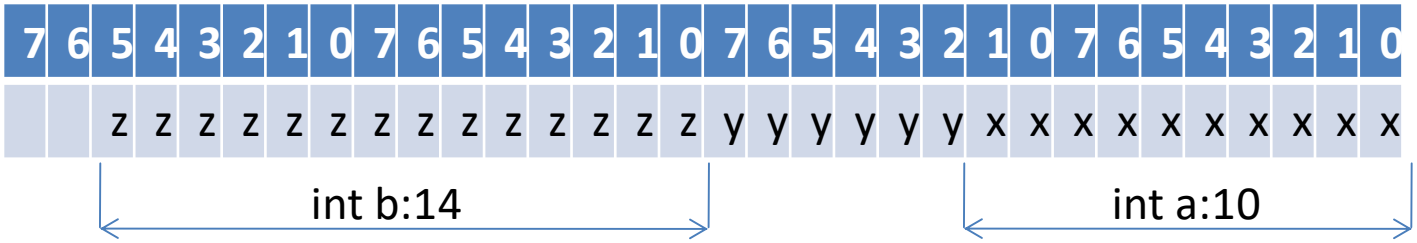




3. Битовые поля

Имя поля может быть опущено для выравнивания полей по плану программиста

```
struct {
        int a:10;
        int :6;
        int b:14;
} y;
```



3. Битовые поля

Пример. Функция вывода битового представления символа.

```
void binar(unsigned char ch){
        union {
                unsigned char uc;
                struct{
                        unsigned a1:1;
                        unsigned a2:1;
                        unsigned a3:1;
                        unsigned a4:1;
                        unsigned a5:1;
                        unsigned a6:1;
                        unsigned a7:1;
                        unsigned a8:1;
                } byte;
        } cod;
        cod.uc = ch;
        cout << "bits:   7 6 5 4 3 2 1 0";
        cout << "\nvalues: " << cod.byte.a8 << ' ';
        cout << cod.byte.a7 << ' ';
        cout << cod.byte.a6 << ' ';
        cout << cod.byte.a5 << ' ';
        cout << cod.byte.a4 << ' ';
        cout << cod.byte.a3 << ' ';
        cout << cod.byte.a2 << ' ';
        cout << cod.byte.a1 << ' ';
}
```

4. Задания

1. Создайте структуру car, хранящую информацию об автомобиле (марка, модель, объём двигателя, год выпуска, цвет и т.п.). В функции main() создайте массив типа car. Занесите в массив и выведите из массива на экран данные о нескольких автомобилях.
2. С помощью битовых полей создайте функцию для вывода шестнадцатеричного кода символа.