Программирование на языке C++
Лекция

Объединения. Перечисления

Объединения

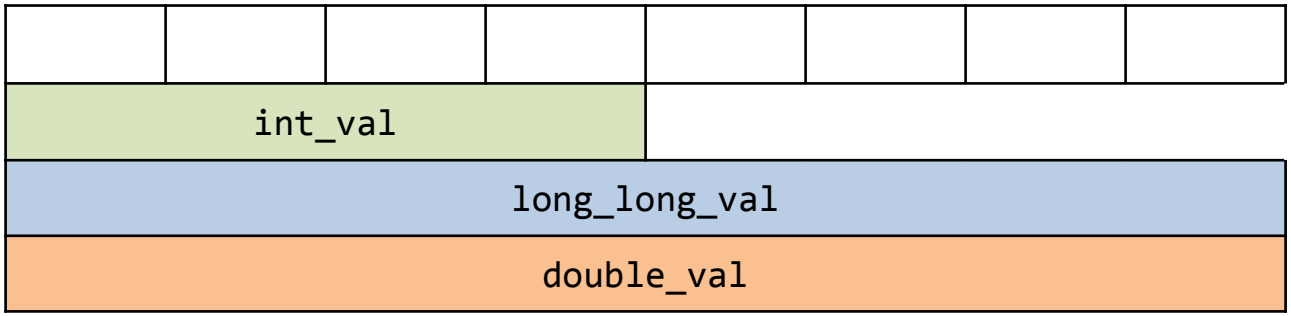
Что такое объединение?

Объединение — это пользовательский тип данных, который может хранить в пределах *одной области* памяти разные типы данных, но в каждый момент времени только один из них.

Размер объединения определяется размером крупнейшего поля.

Объявление

```
union one4all {
    int int_val;
    long long long_long_val;
    double double_val;
};
```



Инициализация

```
union one4all {
    int int_val;
    long long long_long_val;
    double double_val;
};

// Выражение вычисляется и присваивается
первому полю в объединении
one4all num = {10.1};
cout << num.int_val; // 10
```

Использование I

```
one4all pail;
pail.int_val = 15;          // сохранение int
cout << pail.int_val;       // 15

pail.double_val = 1.38;     // сохранение double
cout << pail.double_val; // 1.38
cout << pail.int_val;       // -515396076
```

Использование II

```
struct widget {
    char brand[20];
    int type; // Определяет что лежит в id_val
    union id {
        long id_num;
        char id_char[20];
    } id_val;
};
```

Анонимные объединения

```
union {                         // Нет имени
        long id_num;
        char id_char[20];
};                              // Нет переменных
```

Две переменные работающие с одной областью памяти

Анонимные объединения

```
int main(){
    union {
        int i;
        double d;
    };
    i = 12345678;
    cout << i << '\n';  // 12345678

    d = 12345678;
    cout << i << '\n';  // -1073741824
}
```

Перечисления

Что такое перечисление

Перечисление – это пользовательский тип данных, определяющий набор целочисленных констант.

Зачем нужен:

- Сделать код более читабельным путём замены «магических чисел» на элементы перечисления;
  Пример: `return 0; return SUCCESS;`

- Как дополнительный контроль, защищающий от случайных, автоматических преобразований типов.

Объявление I

```
enum Color {
// Элементы перечисления называются перечислителями
// Они определяют все допустимые значения данного типа
COLOR_BLACK,   // Перечислители разделяются запятыми
COLOR_RED,     // Обычно они пишутся заглавными буквами
COLOR_BLUE,    // но это не обязательно
COLOR_GREEN,
COLOR_WHITE,
COLOR_CYAN,
COLOR_YELLOW,
COLOR_MAGENTA, // В C++11 можно ставить запятую в конце
};             // Точка с запятой обязательна
```

Объявление II

```
enum Color {
COLOR_BLACK,        // Присваивается целое значение 0
COLOR_RED,          // 1
COLOR_BLUE = 7,     // Можно присвоить своё значение
COLOR_GREEN,        // 8 Нумерация продолжается
COLOR_WHITE = 7,    // Можно дублировать значения
COLOR_CYAN,         // 8
COLOR_YELLOW,       // 9
COLOR_MAGENTA = -1 // отрицательные тоже допускаются
};
```

Объявление III

```
enum Color {
  YELLOW,
  BLACK, // имя BLACK теперь занято
  PINK
};

enum Feelings {
  SAD,
  ANGRY,
  BLACK // ошибка, BLACK уже использован в Colors
};

int BLACK = 3; // ошибка
```

Переменные

```
enum Color {
  YELLOW,
  BLACK,
  PINK
} a;        // Создание во время объявления

enum {
  ONE,
  TWO,
  THREE
} b;        // Создание из анонимного перечисления

Color c;  // Обычным способом
```

Инициализация / Присваивание

```
enum Color {
  YELLOW,
  BLACK,
  PINK
};

Color c = YELLOW;
Color pig(PINK);
Color zebr = Color::BLACK;
Color window = 0;  // Ошибка
```

Ввод / Вывод

```
enum Color {
  YELLOW,
  BLACK,
  PINK
};

Color pig(PINK);
cout << pig;      // Преобразуется в число (2)
cin >> pig;       // Ошибка компиляции

int input;
cin >> input;
pig = static_cast<Color>(input);
```

Операции I

```
enum Color { YELLOW, BLACK, PINK };
```

Перечисления преобразуются в целое число автоматически:

```
Colors c = BLACK;
int i = 5 + c;       // i = 5 + 1;
int j = 5 + PINK;    // j = 5 + 2;
```

Переменной перечисляемого типа можно присвоить только перечислитель соответствующего типа:

```
Colors dor = YELLOW;
c = dor;
c = 0;  // Ошибка   c != YELLOW
c = static_cast<Color>(0); // Явное преобразование можно
```

Операции II

```
enum Color { YELLOW, BLACK, PINK };
Colors c = BLACK, pig = PINK;
```

Переменные перечисляемого типа часто используются в:

- Операторах ветвления:
  ```
  if (pig == PINK) ...;
  switch(c){
      case YELLOW : ...; break;
      case BLACK : ...; break;
  }
  ```
- В качестве возвращаемого значения:
  ```
  return ERROR_OPENING_FILE;
  return SUCCESS;
  ```

Преобразования в перечисление

```
enum Color { YELLOW, BLACK, PINK = 10 };
Colors c = BLACK;

c = static_cast<Color>(0);
c = Color(0);     // в стиле Си
c = (Color) 0;    // в стиле Си

// Допускается, но поведение будет не определено
c = static_cast<Color>(5);
```