Лекция: Типы данных в C++

Типы данных — это основа любого языка программирования, включая C++. Они определяют, какие данные могут храниться в переменных, какие операции можно производить над этими данными и сколько памяти они занимают.

1. Основные типы данных (Primitive Types)

C++ поддерживает множество примитивных типов данных, каждый из которых предназначен для работы с конкретным набором значений.

a) Целые числа (Integer Types)

Используются для хранения целых чисел, без дробной части.

- `int` — базовый тип данных для целых чисел.

```
int x = 10; // переменная x хранит целое число 10
```

  Размер: обычно 4 байта (но зависит от платформы), диапазон значений: -2^31 до 2^31-1.

- `short` и `long` — варианты для меньших или больших целых чисел.

```
short s = 1000;    // обычно 2 байта
long l = 100000L;  // обычно 4 или 8 байт в зависимости от архитектуры
```

- `unsigned` — модификатор, который указывает на хранение только положительных чисел.

```
unsigned int u = 42; // беззнаковое целое число, только положительные
значения
```

b) Вещественные числа (Floating-Point Types)

Используются для хранения чисел с плавающей точкой (дробные числа).

- `float` — тип для чисел с одинарной точностью.

```
float pi = 3.14f; // занимает 4 байта
```

- `double` — для чисел с двойной точностью.

```
double g = 9.81;  // занимает 8 байт
```

- `long double` — для чисел с ещё большей точностью, чем `double`. Размер может варьироваться в зависимости от компилятора.

c) Символьный тип (Character Type)

Используется для хранения символов.

- `char` — тип для хранения символов в кодировке ASCII.

```
char letter = 'A'; // занимает 1 байт
```

Также можно использовать для хранения небольших целых чисел, так как каждый символ в ASCII представлен числовым значением.

d) Логический тип (Boolean Type)

Для хранения значений логического типа.

- `bool` — тип для хранения значений истина (true) или ложь (false).

```
bool isHappy = true; // занимает 1 байт (или 1 бит в некоторых компиляторах)
```

2. Модификаторы типов (Type Modifiers)

В C++ можно изменять поведение типов данных с помощью модификаторов.

- `signed` / `unsigned` — указывает, могут ли числа быть отрицательными или только положительными.

```
signed int a = -5;   // знаковый тип
unsigned int b = 5;  // беззнаковый тип
```

- `short` / `long` — изменяют размер типа данных.

```
short int si = 100;  // обычно 2 байта
long int li = 1000;  // обычно 4 или 8 байт
```

3. Пользовательские типы данных

Кроме базовых типов данных, в C++ можно создавать пользовательские типы данных с помощью таких конструкций, как структуры, перечисления и классы.

a) Перечисления (Enumerations)

Тип данных, позволяющий задавать набор именованных целочисленных значений.

```
enum Color { RED, GREEN, BLUE };
Color myColor = RED;
```

b) Структуры (Structures)

Позволяют объединять различные типы данных в одном объекте.

```
struct Point {
    int x;
    int y;
};
Point p1 = {10, 20}; // структура с двумя полями
```

c) Классы (Classes)

Классы — это расширенные структуры с дополнительными возможностями, такими как методы, инкапсуляция и наследование.

```
class Rectangle {
    public:
        int width, height;
        int area() {
            return width * height;
        }
};
Rectangle rect = {10, 5};
int area = rect.area(); // вызов метода
```

4. Указатели и ссылки

Указатели и ссылки играют важную роль в управлении памятью и передаче данных в C++.

a) Указатели (Pointers)

Указатели — это переменные, которые хранят адреса других переменных.

```
int a = 10;
int *ptr = &a;  // указатель на переменную a
```

b) Ссылки (References)

Ссылки — это альтернативный способ работы с переменными, позволяющий обращаться к ним по ссылке.

```
int b = 20;
int &ref = b;  // ссылка на переменную b
```

5. Константы (Constants)

В C++ можно объявить переменные, которые нельзя изменить после инициализации.

- `const` — модификатор, который делает переменную неизменной.

  ```
  const int pi = 3.14;  // константа
  ```

- `constexpr` — для выражений, которые могут быть вычислены на этапе компиляции.

  ```
  constexpr int max_value = 100;
  ```

6. Тип `auto`

C++ поддерживает автоматическое определение типов переменных с помощью ключевого слова `auto`. Компилятор автоматически определит тип данных на основе присвоенного значения.

```
auto x = 5;    // x будет иметь тип int
auto y = 3.14; // y будет иметь тип double
```

7. Приведение типов (Type Casting)

Иногда нужно преобразовать одну переменную в другой тип. В C++ существуют следующие способы приведения типов:

- **Явное приведение (C-style cast)**:

  ```
  int a = 5;
  double b = (double)a;
  ```

- **C++ приведение** (например, `static_cast`):

  ```
  ```

```cpp
int x = 10;
double y = static_cast<double>(x);
```

Заключение

Типы данных являются фундаментом любой программы, написанной на C++. От выбора типа данных зависит, как эффективно будет использоваться память и производительность программы. Понимание типов данных и их правильное использование помогает писать надёжный и оптимизированный код.