Занятие 1:
переменные, типы данных, управляющие конструкции

- Переменные
- Основные типы данных
- Операции
- Выражения
- Явное и неявное приведение типов
- Арифметические операции и выражения
- Логические операции и выражения
- Консольный ввод-вывод
- Условный оператор if
- Оператор выбора switch
- Операторы циклов for, while, do-while
- Операторы break и continue

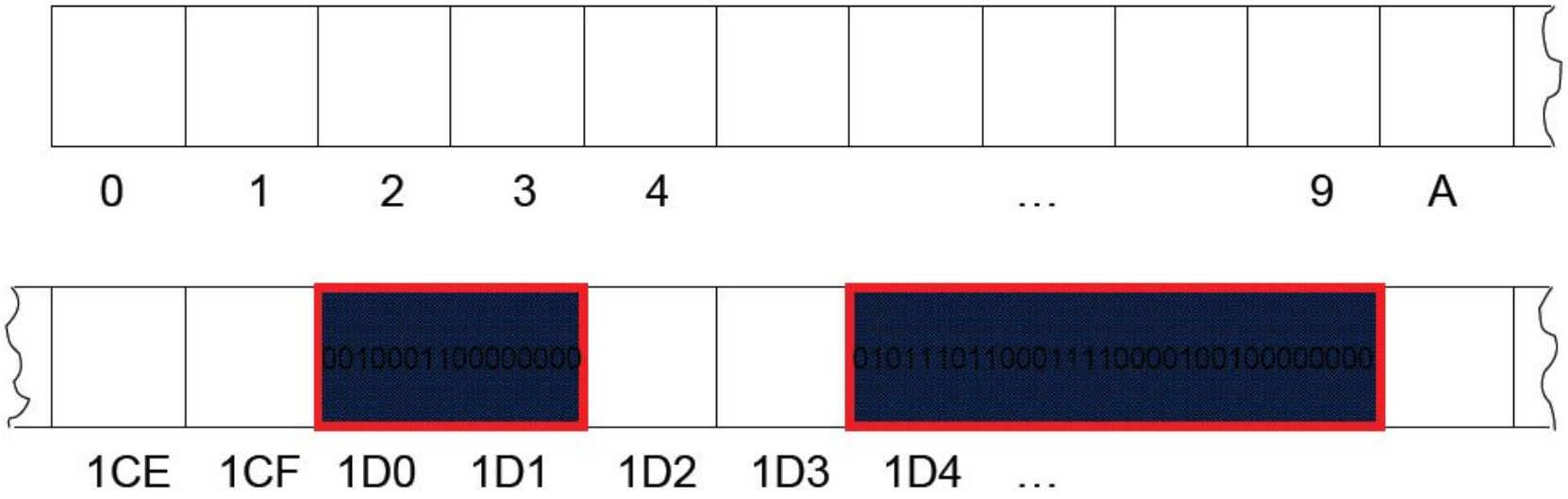
Переменная

– это область в *памяти*, содержащая *данные*, к которой можно *обращаться* по имени или другим способом.

Свойства переменной:

- Имя*
- Адрес
- Тип
- Значение

Модель виртуальной памяти



Age = 35

Тип переменной

Определяет:

- Размер (в байтах)
- Диапазон значений
- Допустимые операции

Основные типы

Целочисленные	С плавающей точкой
• bool • char • short int • int • long int • wchar_t	• float • double • long double
• signed char • unsigned char • unsigned short int • unsigned int • unsigned long int	

Типичные размеры типов C++

Тип данных	Размер (байт)	Диапазон значений
bool	1	false, true
signed char	1	-128 … +127
unsigned char	1	0 … +255
short int	2	-32 768 … +32 767
unsigned short int	2	0 … +65 535
int, long int	4	-2^{31} … $+(2^{31} - 1)$
unsigned int	4	0 … $+(2^{32} - 1)$
wchar_t	2	
float	4	$-3.4*10^{38}$ … $+3.4*10^{38}$
double, long double	8	$-3.4*10^{308}$ … $+3.4*10^{308}$

Размеры типов удовлетворяют соотношениям:

1. *sizeof* (char) = 1
2. *sizeof* (char) ≤ *sizeof* (short int) ≤ *sizeof* (int) ≤ *sizeof* (long int)
3. *sizeof* (float) ≤ *sizeof* (double) ≤ *sizeof* (long double)

Объявление переменных

[спецификатор-класса-памяти] спецификатор-типа описатель
[=инициатор] [,описатель [= инициатор]]... ;

Примеры объявления переменных:

```
char a; // объявление непроинициализированной переменной
int b = 10; // объявление переменной с инициализацией


// очень сложное и некрасивое объявление
// сразу нескольких переменных
static unsigned long int c = b * 5,
    array[10] = {1,2,3}, **pptr;
```

Объявление константных переменных

Ключевое слово **const** при объявлении:

```
const char a;  // Ошибка! Константную переменную надо
// инициализировать при объявлении

const int b = 10;  // Нет ошибки

b = 20;  // Ошибка! Константную переменную нельзя изменить
```

Литералы

Тип	Примеры
`char`	`'Q', '0', '\'', '\\', '\n', '\r', '\t', '\x20'`
`int`	`0  -1  1234`
`unsigned int`	`4000000000  1234u`
`long int`	`4000000000L`
`unsigned long int`	`4000000000UL`
`double`	`0.0  0.  .0  12e-3  1.2E+4`
`float`	`1.2e3F`
`const char*`	`"I love C++"`

Стандартный вывод

• **cout** – объект для вывода данных в стандартный выходной поток

• **cerr** – для вывода данных в стандартный поток ошибок

• Необходимо подключить заголовочный файл **<iostream>**

• Необходимо либо использовать директиву **using namespace std;** либо предварять каждое появление объектов **cout**, **endl** и др. префиксом **std::**

• Для вывода данных используется операция вывода в поток **<<**

Манипуляторы потоков вывода

Манипулятор	Назначение
`endl`	перевод на новую строку + сброс потока
`setw`	устанавливает ширину поля для вывода
`oct`	вывод целых чисел в восьмеричной системе
`dec`	вывод целых чисел в десятичной системе
`hex`	вывод целых чисел в шестнадцатеричной системе
`setprecision`	точность вывода вещественных чисел

! Для манипуляторов с параметрами требуется подключить **<iomanip>**

Стандартный ввод

- **`cin`** – объект для ввода данных из стандартного входного потока.
- Определён в заголовочном файле **`<iostream>`**
- Для ввода используется операция **`>>`** и набор других методов
- Ввод в C++ безопасен по типу данных
- По умолчанию операция **`>>`** игнорирует пробельные символы. Поведение изменяется с помощью манипуляторов потока **`skipws/noskipws`**

Операции

• Пример: арифметические операции **`+, -, *, /, %`**

• Это специальный способ записи некоторых действий
Сравните: **`sum(a, b)`** и **`a + b`**

• Арность операции – количество операндов:
- унарные: **`-b`**
- бинарные: **`b * c`**
- тернарные: **`b ? c : d`**

• Почти все операции имеют возвращаемое значение

• Некоторые операции имеют побочные эффекты – изменение операндов в процессе выполнения операции

Выражения

• Это сочетание переменных, литералов, операций и функций, удовлетворяющее систаксису языка

• Пример: `a + 2 * b * sin(c)`

• Операции в выражении вычисляются в соответствии с их *приоритетом* и *ассоциативностью*, которые можно изменить с помощью круглых скобок

• Выражения, как и операции, возвращают некоторое значение

Целочисленное деление и взятие остатка от деления

1) Выделение полного количества и остатка

```
int kop = 378; // число копеек
int rub = k / 100; // число полных рублей
int kopLeft = kop % 100; // останется копеек после
// изъятия всех полных рублей
```

2) Делится ли число нацело на другое число: Если **`n % k`** равно **`0`**, то **`n`** делится на **`k`** без остатка

3) Отображение больших чисел в небольшой диапазон
Пример.

Диапазон чисел [0 … 4294967295]. Желаемый диапазон [0 … 49].

```
150 % 50 = 0
2147483388 % 50 = 38
```

Операции модификации с присваиванием

```
a = a + 2;      <=>    a += 2;
```

- `+=` прибавление с присваиванием
- `-=` вычитание с присваиванием
- `*=` умножение с присваиванием
- `/=` деление с присваиванием
- `%=` взятие остатка с присваиванием

Примеры	Эквивалентная запись
`d *= 2.0 * PI;`	`d = d * 2.0*PI;`
`b %= 3;`	`b = b % 3;`

Операции инкремента и декремента

- `++x` – преинкремент
- `x++` – постинкремент
- `--x` – предекремент
- `x--` – постдекремент

Все варианты увеличивают/уменьшают значение переменной на 1.
Отличаются возвращаемым значением:
- «пре»-варианты возвращают новое значение
- «пост»-варианты возвращают старое значение

Операция `sizeof`

1. Позволяет определить размер в байтах:

• типа данных:

```
sizeof(long double)
```

• переменной:

```
sizeof(a)
```

• выражения:

```
sizeof(a + b/c)
```

2. Не является функцией

3. Вычисляется на этапе компиляции

4. Рекомендуется использовать для переносимости

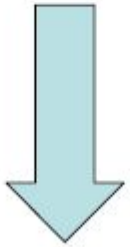
Неявное приведение типов

Пример преобразования без потерь:

- **char ➔ int**

```
char x = 'X';
int z;
z = x; // x = 88
```

00000000 00000000 00000000 **01011000** **char x;**



00000000 0000000 00000000 01011000 **int z;**

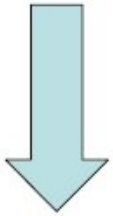
Неявное приведение типов

Пример преобразования без потерь:

- **char ➔ int**

```
signed char x = -40;
int z;
z = x; // z = -40
```

11111111 11111111 11111111 **11011000** **signed char x;**



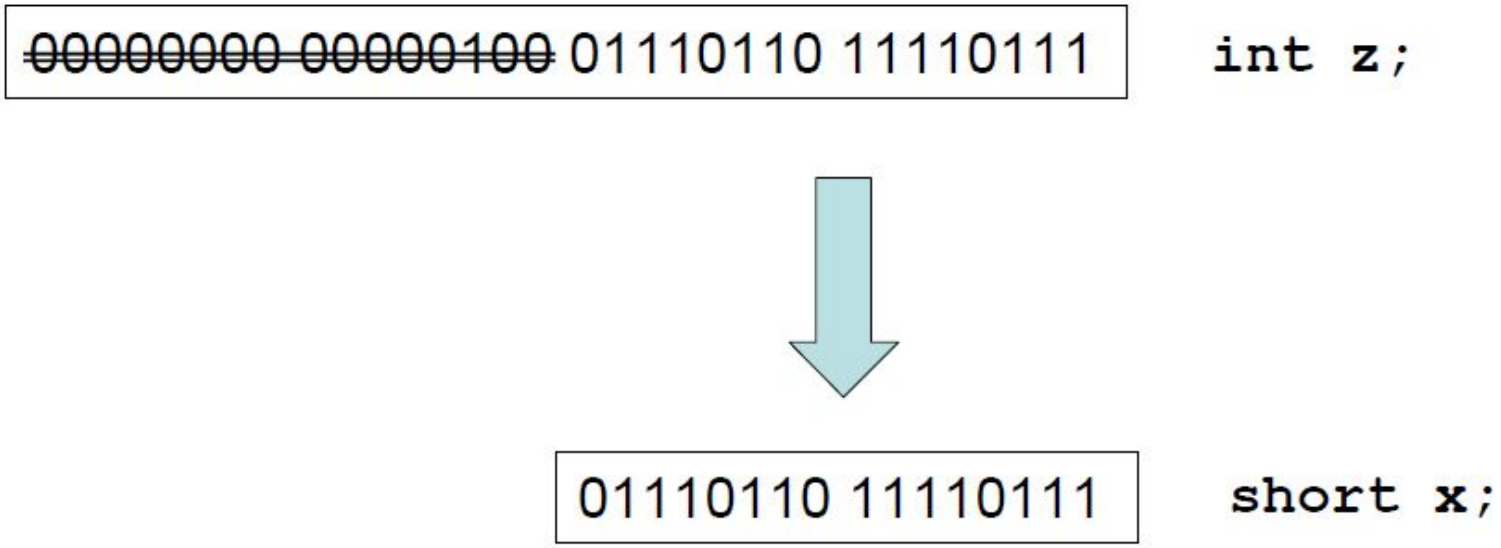
11111111 11111111 11111111 11011000 **int z;**

Неявное приведение типов

Пример преобразования с потерей данных:

- **int ➔ short**

```
int z = 292599;
short x = z; // x = 30455
```



Неявное приведение типов

Пример преобразования с потерей данных:

- **float ➔ int**

```
float f = -2.99;
int x = f; // x = -2

float f2 = 2.12e+15;
int x2 =  f2; // x2 - несуразный результат
```

- **int ➔ float ➔ int**

```
int x = 1234567899;
float f = x; // f = 1.234567936e+9
int y = f;   // y = 1234567936
```

Неявное приведение типов в арифметических выражениях

```
char c = 5;
int  i = 10;
long L = 203930;
float f = -3.2;
double d = 2.5;
```

```
c + i;            (c: char → int)
c + c;            (c: char → int)
i + L;            (i:  int → long)
f + i;            (i: int → float)
d + i - c + f;    (i: int → double,
                   c: char → double,
                   f: float → double)
```

Неявное приведение типов в арифметических выражениях

Условное старшинство базовых типов в арифметических операциях:

```
char, short int, bool
     ➔ int
        ➔ unsigned int
           ➔ long
              ➔ unsigned long
                 ➔ float
                    ➔ double
                       ➔ long double
```

Операция явного приведения типов

Синтаксис:

- **`(тип)выражение`**

- приведение в стиле языка C

- **`тип(выражение)`**

- приведение в стиле вызова конструктора типа

- **`static_cast<тип>(выражение)`**

- специфичное для C++ приведение типов

Управляющие конструкции

- Условный оператор **`if`**
- Оператор выбора **`switch`**
- Операторы циклов

`for`
`while`
`do-while`

Оператор `if`

```
if (условие_1)
   операторы_1
else if (условие_2)
   операторы_2
else if (условие_3)
   операторы_3
…
else
   операторы_n
```

Условие в операторе

- выражение типа `bool`
- выражения с операциями отношения:
  `>`
  `>=`
  `<`
  `<=`
  `==` (равно)
  `!=` (не равно)
- выражение с логическими операциями
  `!` (логическое отрицание)
  `&&` (логическое умножение, логическое «и»)
  `||` (логическое сложение, логическое «или»)
- выражение другого типа, который может быть приведён к `bool`

Оператор **switch**

```
switch (выражение)
{
  case значение_1:
    операторы_1
  case значение_2:
    операторы_2
  …
  default:
    операторы_n
}
```

! Для правильной работы **switch** в конце каждой группы операторов обычно нужен оператор **break**

Оператор for

```
for (инициализация; условие; действие)
  тело_цикла
```

инициализация – выполняется один раз в начале цикла

условие – проверяется перед каждой итерацией цикла. Если условие истинно, цикл продолжается

действие – выполняется в конце каждой итерации

тело_цикла – один оператор или блок операторов

Обычно цикл **for** используется, когда заранее известно число итераций

Оператор **while**

```
while (условие)
   тело_цикла
```

условие – проверяется перед каждой итерацией цикла. Если условие истинно, цикл продолжается

тело_цикла – один оператор или блок операторов

Обычно цикл **while** используется, когда заранее неизвестно число итераций, а условие нужно проверять перед следующей итерацией

Оператор do-while

```
do
   тело_цикла
while (условие);
```

условие – проверяется после каждой итерации цикла. Если условие истинно, цикл продолжается

тело_цикла – один оператор или блок операторов

Обычно цикл **do-while** используется, когда заранее неизвестно число итераций, а условие нужно проверять после выполненной итерации

Операторы перехода

- **`break`** – прерывание цикла или выход из **`switch`**
- **`continue`** – переход к следующей итерации цикла
- **`goto`** – переход по метке