О программировании в общем и о C++ в частности. Лекция I

Поспелов Евгений Анатольевич

6 октября 2016 г.

Мотивация (физика)

- Простых задач не осталось
- Эксперимент на сложных системах - дорог, а вычислительные мощности сравнительно дёшевы
- Основными языками для работы на суперкомпьютерах остаются: C, C++ и Fortran

Историческая справка

- Деннис Ритчи, Bell Labs: 1973 - публикация языка C

- Разработан для упрощения написания операционных систем
- 1989 - первый стандарт ANSI C89 (ISO C90)

Историческая справка

- Опять Bell Labs, Бьярне Страуструп (Bjarne Straustrup)
- 1985, первый коммерческий релиз языка C++

- 1998 - первый стандарт ISO C++98. На данный момент выпущены C++03, **C++11**, C++14

Информация

Литература

- Алекс Эллайн, "C++. От ламера до программера 2015 г., Питер ("Jumping into C++")
- Татьяна Павловская, "C/C++. Процедурное и объектно-ориентированное программирование 2014 г., Питер
- Питер Готшлинг, "Современный C++ для программистов, инженеров и учёных 2016 г., Вильямс ("In Discovering Modern C++")

Альтернатива онлайн

- http://ru.cppreference.com/w/cpp
- http://www.cplusplus.com/reference/
- http://www.cprogramming.com/
- https://github.com/posgen/OmsuMaterials

Характеристики C++

C++ является:

- компилируемым
- статически типизированным
- мультипарадигменным (поддерживает процедурное, объектно-ориентированное и обобщённое программирование)

Какие особенности определяют язык программирования?

Синтаксические отличия:

1. Типы данных и объявления переменных для них
2. Основные конструкции управления ходом выполнения программы (ветвление, циклы, безусловные переходы)
3. Способ определения замкнутых, повторно используемых блоков кода (функции)
4. Возможность создания пользовательских типов данных

Общий вид объявления переменной:

```
[указания компилятору] <тип_данных> <имя_переменной>[начальное значение]<;>
```

, где треугольные скобки означают обязательные части, квадратные - опциональные.

Коментарии - произвольный текст в файле с исходным кодом, который игнорируется компилятором и не влияет на ход программы.

```
1 // Это однострочный комментарий
2
3 /*
4 Пример
5   многострочного
6     комментария
7 */
```

Таблица 1 : Целочисленные встроенные типы данных в C++

Тип	Принимаемые значения
short	короткие целые числа со знаком
int	целые числа со знаком
unsigned	целые числа без знаком
size_t	тоже самое, что и unsigned
long	длинные целые числа со знаком

Объявление переменных

```
1 // Переменные конкретного типа можно ↩
      определять по одной
2 size_t counter;
3 size_t array_size;
4
5 // А можно и несколько подряд, через запятую
6 int width, height, area;
```

Присваивание переменным значений

```
1  size_t counter;
2  // Присвоение переменной начального значения
3  counter = 5;
4
5  /*
6    Присвоение допустимо при объявлениии ↩
       переменной.
7    В этом случае, говорят об её определении (↩
       или инициализации)
8  */
9  int width = 6, height = 1, area, karma;
10 area = width;
11 karma = -88;
```

Присваивание переменным значений

```
1 // Альтернативная инициализация с помощью фигурных скобок
2 size_t width{7};
3
4 // Так можно, дробная часть отбрасывается:
5 size_t height = 5.65;
6 // А так – лучше не делать, поскольку типы не совпадают
7 size_t max_limit{6.78}; // Предупреждение или ошибка времени компиляции
```

Арифметические операции

```
1  int balance, rate, total;
2
3  balance = 4 + 6;
4  rate = balance - 8;
5  total = 2 * balance + (6-2);
6
7  // Целочисленное деление - дробная часть ↩
       отсекается
8  rate = 7 / 2; // rate равен 3
9
10 // Взятие остатка от деления
11 rate = 7 % 2; // rate равен 1
```

Арифметические операции

```
1  int balance{5}, rate{10}, total{15};
2
3  balance = balance + 1;
4  // Сокращённая запись
5  balance += 1;
6  // Определены и остальные арифметические ↩
       операции: -=, *=, /=, %=
7
8  // Инкремент/Декремент - увеличение/↩
       уменьшение значения переменной на единицу
9  rate++; // rate стал равным 11
10 ++rate; // -//- 12
11
12 rate--; // снова 11
13 --rate; // теперь 10
```

Арифметические операции

```
1  // Разница пре– и пост– инкремента
2  int balance{5}, total{5};
3
4  std::cout << balance++; // напечатает на ↩
       экране 5
5  // значение balance равно 6
6
7  std::cout << ++total; // напечатает 6
8  // значение total равно 6
9
10 int number{-6};
11 +number;
12 -number;
```

Побитовые операции

```
1  unsigned number{4}, next_number;
2
3  // Побитовый сдвиг вправо на n позиций
4  next_number = number << 2;
5  // Побитовый сдвиг влево на n позиций
6  next_number = number >> 3;
7
8  // Побитовое "И"
9  next_number = number & 2;
10 // Побитовое "ИЛИ"
11 next_number = number | 3;
12
13 next_number >>= 1;
```

float - действительное число одинарной точности (≈ 7-8 знаков после запятой)

double - действительное число двойной точности (≈ 15 знаков после запятой)

```
1 double speed{5.5}, distance;
2 distance = speed / 3.0;
3
4 speed *= 4;
```

bool - логический (булев) тип данных, принимающий только два значения - **true** иди **false**

```
1  bool truth = true, falsey = false, result;
2
3  // Логическое "И"
4  result = truth && falsey; // result равен ↩
       false
5  // Логическое "ИЛИ"
6  result = truth || falsey; // result равен ↩
       true
7
8  // Отрицание:
9  result = !falsey; // result равен true
10
11 // Совместимость с C
12 result = 0; // result равен false
```

```
1 bool var1, var2;
```

Таблица истинности

	var1	var2	Результат
&&	true	true	true
&&	true	false	false
&&	false	true	false
&&	false	false	false
||	true	true	true
||	true	false	true
||	false	true	true
||	false	false	false

char - тип данных для хранения одиночных символов: ’a’, ’b’, ’d’, ’4’, ’%’, ”. Как правило, размер ограничен 1 байтом.

```
1 char symbol = '%';
2 std::cout << symbol; // печатаем знак ↩
    процента
3
4 symbol = '5';
5 symbol += 2; // допустимая операция, теперь ↩
    symbol равен '7'
6
7 // Набор специальных символов: \n, \t, \\, \↩
    a
8 symbol = '\n'; // знак переноса строки
```

Область видимости переменной - та часть программы, как правило - файл, которая имеет доступ к переменной. В C++ определяется набором фигурных скобок - {}

```
1  char symbol = 's';
2
3  {
4    int weight = 55;
5    // здесь можем обращаться к обеим ↩
       переменным — symbol и weight
6    std::cout << weight;
7    std::cout << symbol;
8  }
9  // Здесь уже невозможно получить доступ к ↩
     weight, ошибка компиляции
10 std::cout << weight;
```

Операция	Оператор
равенство	==
неравенство	!=
больше	>
меньше	<
больше или равно	>=
меньше или равно	<=

Все операторы сравнения в C++ возращают **bool**

```
1 int width = 5, height = 4;
2
3 bool is_equals;
4 // is_equals равен false
5 is_equals = width == height;
```

Осторожно

Сравнение действительных чисел может быть неожиданным

```
1  double first_rate, second_rate;
2  //Пусть переменная first_rate была заполнена значением 5.5 из файла
3  second_rate = 2.75 * 2;
4
5  bool is_equals;
6
7  // Здесь ок, результат будет равен true
8  is_equals = first_rate == 5.5;
9
10 // Здесь опасность, результат может быть false
11 is_equals = second_rate == 5.5;
```

Общий синтаксис

```
if (<логическое выражение>) {
  <набор инструкций>
} [else {
  <набор инструкций>
}]
```

```
1 int max_score{3}, min_score{5};
2
3 if ( max_score < min_score )
4     std::cout << "Как-то странно\n";
5 std::cout << "эта инструкция не относится к ↩
    оператору ветвления";
```

```
1  int max_score{3}, min_score{5};
2  // Так гораздо понятнее:
3  if ( max_score < min_score ) {
4      std::cout << "Как-то странно\n";
5  }
6  std::cout << "эта инструкция не относится к ↩
       оператору ветвления";
7
8  int current_score = 4;
9  if ( (current_score >= min_score) && (↩
       current_score <= max_score) ) {
10     std::cout << "Всё ок";
11 } else {
12     std::cout << "Какая-то аномалия";
13 }
```

```
1  int current_score = 7;
2
3  if ( current_score == 4 ) {
4      current_score += 4;
5  } else if ( current_score == 5 ) {
6      current_score += 1;
7  } else if ( current_score == 6 ) {
8      current_score -= 10;
9  } else {
10     current_score = 0;
11 }
```

Оператор **?:**

```
<логическое выражение>
    <?> <выражение, если истинно>
    <:> <выражение, когда ложно>;
```

```
1 int max_score{13}, min_score{5};
2
3 int result;
4 result = ( (max_score - min_score) < 5 ) ? ↩
    min_score * 2 : max_score / 2; )
```

Конструкция **switch**

```
switch (<выражение>) {
    case <значение_1>:
        [инструкции]
        <break>;
    case <значение_2>:
        [инструкции]
        <break>;
    ...
    case <значение_T>:
        [инструкции]
        <break>;
    [default:
        [инструкции]
    ]
}
```

```
1  int rate, score;
2
3  // Вычисляем rate
4
5  switch ( rate ) {
6    case 2:
7      score = 1; break;
8    case 5:
9      score = 2;
10     break;
11   case 8:
12     score = 3;
13     break;
14   default:
15     score = 5;
16 }
```

Цикл

Под **циклом** в программировании понимается обособленный набор повторяющихся N раз инструкций. Причём, $N \geq 0$

Дополнительные определения

- **Тело цикла** - набор повторяющихся инструкций
- **Итерация** (или шаг цикла) - один проход по телу цикла (выполнение всех описанных инструкций)

Циклы **while** и **do ... while**

```
while (<логическое выражение>) {
   [инструкции]
}

do {
   [инструкции]
} while (<логическое выражение>);
```

Цикл **while**

```
1 int counter{10};
2
3 while ( counter > 0 ) {
4   std::cout << counter * counter << "\n";
5   counter--;
6 }
```

Цикл **do ... while**

```
1 int counter{0};
2
3 do {
4   std::cout << counter * counter << "\n";
5   counter--;
6 } while ( counter < 10 );
```

Бесконечный цикл на примере **while**

```
1 while( true ) {
2   std::cout << "*";
3 }
```

Цикл **for**

```
for (   [инициализация]<;>
        [условие выхода]<;>
        [итеративное изменение]
    ) {
  [инструкции]
}
```

Цикл **for**

```
1 // Бесконечный цикл
2 for (;;) {
3   // что-то полезное
4 }
5
6 // Вывод значений квадратного корня для ↩
    чисел от 0 до 9
7 for (int i = 0; i < 10; i++) {
8   std::cout << std::sqrt(i) << "\n";
9 }
```

Управление циклами **continue** и **break**

```
1  for (int counter = 0; ; ++counter) {
2    if ( (counter % 2) == 0 ) {
3      std::cout << counter << std::endl;
4      continue;
5    }
6
7    if ( (counter > 15) ) {
8      break;
9    }
10
11 }
```