Программирование на языке C++

Вводный курс

Александр Морозов
gelu.speculum@gmail.com

ИТМО, весенний семестр 2022

Содержание

Составляющие функции

```
1    int main(int argc, char ** argv)
2    {
3      return 0;
4    }
```

Составляющие функции

```
1       int main(int argc, char ** argv)
2       {
3         return 0;
4       }
```

Составляющие функции

```
1       int main(int argc, char ** argv)
2       {
3         return 0;
4       }
```

Составляющие функции

```
1       int main(int argc, char ** argv)
2       {
3         return 0;
4       }
```

Вызов функций

```
1   void f() { }
2
3   void g(int, char, double = 0.1) { }
4
5   int h(int a = -1)
6   {
7     g(a, 'x'); // third parameter is 0.1
8     return a + 2;
9   }
10
11  int main()
12  {
13    f();
14    g(h(), 'a', 0.5);
15  }
```

Вызов функций

```
1   void f() { }
2
3   void g(int, char, double = 0.1) { }
4
5   int h(int a = -1)
6   {
7     g(a, 'x'); // third parameter is 0.1
8     return a + 2;
9   }
10
11  int main()
12  {
13    f();
14    g(h(), 'a', 0.5);
15  }
```

Вызов функций

```
1   void f() { }
2
3   void g(int, char, double = 0.1) { }
4
5   int h(int a = -1)
6   {
7     g(a, 'x'); // third parameter is 0.1
8     return a + 2;
9   }
10
11  int main()
12  {
13    f();
14    g(h(), 'a', 0.5);
15  }
```

Вызов функций

```
1   void f() { }
2
3   void g(int, char, double = 0.1) { }
4
5   int h(int a = -1)
6   {
7     g(a, 'x'); // third parameter is 0.1
8     return a + 2;
9   }
10
11  int main()
12  {
13    f();
14    g(h(), 'a', 0.5);
15  }
```

Блоки и объявления

```
1   int main(int argc, char ** argv)
2   {
3     int a;
4     {
5       int b = a;
6       {
7         int a, b = 101;
8       }
9     }
10    {
11      long a, b = 1L, c = -5;
12      char d = 'X';
13    }
14  }
```

Блоки и объявления

```
1   int main(int argc, char ** argv)
2   {
3     int a;
4     {
5       int b = a;
6       {
7         int a, b = 101;
8       }
9     }
10    {
11      long a, b = 1L, c = -5;
12      char d = 'X';
13    }
14  }
```

Блоки и объявления

```
1   int main(int argc, char ** argv)
2   {
3     int a;
4     {
5       int b = a;
6       {
7         int a, b = 101;
8       }
9     }
10    {
11      long a, b = 1L, c = -5;
12      char d = 'X';
13    }
14  }
```

Блоки и объявления

```
1   int main(int argc, char ** argv)
2   {
3     int a;
4     {
5       int b = a;
6       {
7         int a, b = 101;
8       }
9     }
10    {
11      long a, b = 1L, c = -5;
12      char d = 'X';
13    }
14  }
```

Блоки и объявления

```
1   int main(int argc, char ** argv)
2   {
3     int a;
4     {
5       int b = a;
6       {
7         int a, b = 101;
8       }
9     }
10    {
11      long a, b = 1L, c = -5;
12      char d = 'X';
13    }
14  }
```

Блоки и объявления

```
1   int main(int argc, char ** argv)
2   {
3     int a;
4     {
5       int b = a;
6       {
7         int a, b = 101;
8       }
9     }
10    {
11      long a, b = 1L, c = -5;
12      char d = 'X';
13    }
14  }
```

Блоки и объявления

```
1   int main(int argc, char ** argv)
2   {
3     int a;
4     {
5       int b = a;
6       {
7         int a, b = 101;
8       }
9     }
10    {
11      long a, b = 1L, c = -5;
12      char d = 'X';
13    }
14  }
```

Блоки и объявления

```
1   int main(int argc, char ** argv)
2   {
3     int a;
4     {
5       int b = a;
6       {
7         int a, b = 101;
8       }
9     }
10    {
11      long a, b = 1L, c = -5;
12      char d = 'X';
13    }
14  }
```

Блоки и объявления

```
1   int main(int argc, char ** argv)
2   {
3     int a;
4     {
5       int b = a;
6       {
7         int a, b = 101;
8       }
9     }
10    {
11      long a, b = 1L, c = -5;
12      char d = 'X';
13    }
14  }
```

Область видимости

```
1     int a = 11;
2
3     void foo()
4     {
5       a++;
6       {
7         int a = a;
8         a *= 3;
9       }
10    }
11
12    int b;
13
14    int main()
15    {
16      foo();
17      return a + b;
18    }
```

1

---

1 https://stackoverflow.com/questions/23415661/has-c-standard-changed-with-respect-to-the-use-of-indeterminate-values

Зависимости внутри одной инструкции объявления

```
1    int main()
2    {
3      int a = 11, b = a + 2;
4    }
```

https://stackoverflow.com/questions/24224115/interdependent-initialization-with-commas

Переменные, значения, объекты

Время жизни объектов

```
1    int a = 11;
2
3    int main()
4    {
5      int b;
6      {
7        int c = 11;
8      }
9      return b;
10   }
```

Типы размещения

- автоматический
- статический
- тред-локальный
- динамический

Идентификаторы

- `[A-Za-z_][A-Za-z0-9_]*`
- совпадающие с ключевыми словами – зарезервированы
- содержащие `__` – зарезервированы
- начинающиеся с `_[A-Z]` – зарезервированы
- начинающиеся с `_` – зарезервированы в глобальном пространстве имён

Составляющие текста программы

- идентификаторы
- числовые литералы
- символьные и строковые литералы
- операторы и прочие символы пунктуации

Имена

Имя – идентифицирующее выражение, связанное с некой программной сущностью через определение.

```
1       int a = 1; // declaration
2
3       int f()
4       {
5           return a; // usage
6       }
```

Использование $\rightarrow$поиск имён $\rightarrow$сущность

Литералы

- булевские `true`, `false`
- целочисленные
- дробные
- символьные `'a'`
- строковые `"Hello\n"`
- `nullptr`

Выражения и операторы

```
1    int main(int argc, char ** argv)
2    {
3      argc++ + ++argc; // UB
4      argc = ++argc * 3;
5      return (1 + 2 * 3) * argv[argc - 1];
6    }
```

Выражения и операторы

```
1    int main(int argc, char ** argv)
2    {
3      argc++ + ++argc; // UB
4      argc = ++argc * 3;
5      return (1 + 2 * 3) * argv[argc - 1];
6    }
```

Выражения и операторы

```
1    int main(int argc, char ** argv)
2    {
3      argc++ + ++argc; // UB
4      argc = ++argc * 3;
5      return (1 + 2 * 3) * argv[argc - 1];
6    }
```

Выражения и операторы

```
1    int main(int argc, char ** argv)
2    {
3      argc++ + ++argc; // UB
4      argc = ++argc * 3;
5      return (1 + 2 * 3) * argv[argc - 1];
6    }
```

Список операторов и их свойства

https://en.cppreference.com/w/cpp/language/operator_precedence

Результат и побочные эффекты

```
1   int f(int a, int b)
2   {
3     return a + b;
4   }
5
6   int main()
7   {
8     int a = 0, b = -13;
9     int c = a++ + ++b;
10    b *= 2;
11    return f(a, b);
12  }
```

Невычисляемый контекст

```
1   double f()
2   {
3     return 0.5;
4   }
5
6   int main()
7   {
8     decltype(f()) a = f();
9     auto b = f();
10    return sizeof(b);
11  }
```

Полное выражение

```
1    int main(int argc, char ** argv)
2    {
3      int a = argc + argc / 2, b = a;
4      decltype(a + 2) c = 3;
5      c += a / b;
6      return a + b;
7    }
```

Константные выражения

```
1    int main()
2    {
3      const std::size_t len = 10;
4      std::array<int, len> xxs;
5    }
```

Временные объекты

```
1    int & f(int & a) { return a; }
2
3    int main(int argc, char ** argv)
4    {
5      return 11 + f(argc * 2);
6    }
```

Порядок исполнения

```
1    int main(int argc, char ** argv)
2    {
3      int a, b = argc * 3;
4      a = argc++ + b;
5      f(a, b);
6      f(a++, a);
7      bool x = a > 5 || b < 3;
8      a = a++ + 2, b = a;
9    }
```

Контекст игнорирования результата

```
1    int main()
2    {
3      int a = 1, b = 2;
4      a + b;
5      return a, b;
6    }
```

Инструкции

```
1   int main(int argc, char ** argv)
2   {
3     int a = argc + 2;
4     a *= 3;
5     if (a < 3) {
6       return 0;
7     }
8     if (a > 5)
9     {
10      a += 4;
11    }
12    else
13      a -= 3;
14    for (int i = 1; i < argc; ++i) {
15      a += argv[i][0];
16      continue;
17      a -= 1;
18    }
19  }
```

for

```
1    int main(int argc, char ** argv)
2    {
3      for (int i = 0, j = 1; i + j < argc; ++i, ++j) {
4        i += 2;
5        j -= 2;
6      }
7    }
```

for

```
1    int main(int argc, char ** argv)
2    {
3      for (int i = 0, j = 1; i + j < argc; ++i, ++j) {
4        i += 2;
5        j -= 2;
6      }
7    }
```

for

```
1    int main(int argc, char ** argv)
2    {
3      for (int i = 0, j = 1; i + j < argc; ++i, ++j) {
4        i += 2;
5        j -= 2;
6      }
7    }
```

Минимальный `for`

```
1    int main()
2    {
3      for (;;)
4          ;
5    }
```

if

```
1   int main(int argc, char ** argv)
2   {
3     if (argc > 3) {
4     }
5     if (int i, j, k; argc < 3) {
6       i = 1;
7       j = 2;
8       k = 3;
9     }
10    else {
11      k = 10;
12    }
13    return i + j; // error
14  }
```

switch

```
1   int main(int argc, char ** argv)
2   {
3     int a = 0, b = 1;
4     switch (argc) {
5       case 1:
6         a += 2;
7         b -= 3;
8       case 2:
9         a *= b;
10        break;
11      case 3:
12        return b;
13      default:
14        a += b * 3;
15    }
16  }
```

Объявления и тело `switch`

```
1   int main(int argc, char ** argv)
2   {
3     switch (argc) {
4       case 1:
5         int a = 1;
6         break;
7       default: // error!
8         return argc;
9     }
10  }
```

Объявления и тело `switch`: правильно

```
1   int main(int argc, char ** argv)
2   {
3     switch (argc) {
4       case 1: {
5         int a = 1;
6         break;
7       }
8       default:
9         return argc;
10    }
11  }
```

Ещё более странные вещи с `switch`

```
1   void g(const std::size_t count, char * to, const char * from)
2   {
3     std::size_t n = (count + 7) / 8;
4     switch (count % 8) {
5     case 0: do { *to = *from++; [[fallthrough]];
6     case 7: *to = *from++; [[fallthrough]];
7     case 6: *to = *from++; [[fallthrough]];
8     case 5: *to = *from++; [[fallthrough]];
9     case 4: *to = *from++; [[fallthrough]];
10    case 3: *to = *from++; [[fallthrough]];
11    case 2: *to = *from++; [[fallthrough]];
12    case 1: *to = *from++;
13            } while (--n);
14    }
15  }
```

https://en.wikipedia.org/wiki/Duff's_device

Содержание

Базовые типы

- `void`
- `std::nullptr_t`
- арифметические
  - дробные
  - интегральные
    - логический `bool`
    - символьные `char`...
    - знаковые целые `int`...
    - беззнаковые целые `unsigned`...

Требования к целым числовым типам

Тип	Минимальное значение	Максимальное значение	ILP32	LP64	LLP64
char			8	8	8
signed char	-127	127	8	8	8
unsigned char	0	255	8	8	8
short short int	-32767	32767	16	16	16
unsigned short unsigned short int	0	65535	16	16	16
int	-32767	32767	32	32	32
unsigned unsigned int	0	65535	32	32	32
long long int	-2147483647	2147483647	32	64	32
unsigned long unsigned long int	0	4294967295	32	64	32
long long long long int	$-2^{63}-1$	$2^{63}-1$	64	64	64
unsigned long long unsigned long long int	0	$2^{64}-1$	64	64	64

Требования к дробным числовым типам

Тип	Минимальное число точно представимых десятичных цифр	Максимально представимое число
float	6	1E+37
double	10	1E+37
long double	10	1E+37

Содержание

Числовые расширения

- `signed char` $\rightarrow$`int`
- `unsigned char` $\rightarrow$`int` или `unsigned int`
- `short` $\rightarrow$`int`
- `unsigned short` $\rightarrow$`int` или `unsigned int`
- `char` – либо как `signed char`, либо как `unsigned char`
- `float` $\rightarrow$`double`

Числовые преобразования

```
1  int main()
2  {
3    int a = true; // 1
4    double b = true; // 1.0
5    float c = b; // 1.0
6    unsigned char d = c; // 1
7    unsigned int e = -1; // 0xFFFFFFFF
8    int f = 1.33; // 1
9  }
```

Стандартные арифметические преобразования

```
1   int main()
2   {
3     int a = 1;
4     unsigned b = 2;
5     long long c = -3;
6     float d = 0.5;
7     double e = -1.33;
8
9     auto x = a + b; // unsigned
10    auto y = b + c; // long long
11    auto z = d + e; // double
12    auto u = a + e; // double;
13  }
```

Явные преобразования

```
1    int main()
2    {
3      long long x = -1;
4      auto y = static_cast<unsigned>(x) + 2; // unsigned
5    }
```

Old-style cast

- `(T)x`
- `T(x)`

- `const_cast<T>(x)`
- `static_cast<T>(x)*`
- `static_cast*` + `const_cast`
- `reinterpret_cast<T>(x)`
- `reinterpret_cast` + `const_cast`