Московский государственный университет им. М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Алгоритмы и алгоритмические языки
Лекция 7

28 сентября 2019 г.

Пример Си-программы

```
/* Solving a quadratic equation */
#include <stdio.h>
#include <math.h>
int main (void) {
  int a, b, c, d;
  /* Input coefficients */
  if (scanf ("%d%d%d", &a, &b, &c) != 3) {
    printf ("Need␣to␣input␣three␣coefficients!\n");
    return 1;
  }
  if (!a) {
    printf ("That's␣not␣quadratic!\n");
    return 1;
  }
 <...>
```

Пример Си-программы

```
<...>
  d = b*b - 4*a*c;
  if (d < 0)
    printf ("No solutions\n");
  else if (d == 0) {
    double db = -b;
    printf ("Solution: %.4f\n", db/(2*a));
  } else {
    double db = -b;
    double dd = sqrt (d);
    printf ("Solution 1: %.4f, solution 2: %.4f\n",
            (db+dd)/(2*a), (db-dd)/(2*a));
  }
  return 0;
}
```

Преобразование типов

При присваивании `a = b`:

- «Широкий» целочисленный тип в «узкий»: отсекаются старшие биты
- Знаковый тип в беззнаковый: знаковый бит «становится» значащим
  `signed char c = -1; /* sizeof(c) == 1 */`
  `((unsigned char) c)` $\to$ `255`
- Плавающий тип в целочисленный: отбрасывается дробная часть
- «Широкий» плавающий тип в «узкий»: округление или усечение числа

Явное приведение типов: `(type) expression`

```
d = ((double) a+b)/2;
```

Приведение типов

Неявное приведение типов: происходит, когда операнды двухместной операции имеют разные типы (6.3.1.8)

- Если один из операндов — `long double`, то и второй преобразуется к `long double` (так же для `double` и `float`)
  `long double + double` → `long double + long double`
  `int + double` → `double + double`
  `float + short` → `float + int` → `float + float`
- Если все значения операнда могут быть представлены в `int`, то операнд преобразуется к `int`, так же и для `unsigned int`
  (англоязычный термин — integer promotion)
  `unsigned short(2) + char(1)` → `int(4) + int(4)`
  `unsigned short(4) + char(1)` → `unsigned int(4) + int(4)`
- Если оба операнда — соответственно знаковых или беззнаковых целых типов, то операнд более «узкого» типа преобразуется к операнду более «широкого» типа
  `int + long` → `long + long`
  `unsigned long long + unsigned` →
  `unsigned long long + unsigned long long`

Приведение типов

- Если операнд беззнакового типа более или так же «широк», чем операнд знакового «узкого» типа, то операнд «узкого» типа преобразуется к операнду «широкого» типа
  `int + unsigned long` → `unsigned long + unsigned long`
  `int(4) / unsigned int(4)` → `unsigned int(4) / unsigned int(4)`
  `/* Неверные значения */`
- Если тип операнда знакового типа может представить все значения типа операнда беззнакового типа, то операнд беззнакового типа преобразуется к операнду знакового типа
  `unsigned int(4) + long(8)` → `long(8) + long(8)`
  `unsigned short + long long` → `long long + long long`
- Оба операнда преобразуются к беззнаковому типу, соответствующему типу операнда знакового типа
  `unsigned int(4)+ long(4)` →
  `unsigned long(4) + unsigned long(4)`
- Числа типа `float` не преобразуются автоматически к `double`

Старшинство операций

Старшинство	Ассоциативность
`! ++ -- + - sizeof (type)`	Справа налево
`* / %`	Слева направо
`+ -`	Слева направо
`== !=`	Слева направо
`&&`	Слева направо
`||`	Слева направо
`= += -= *= /= %=`	Справа налево
`,`	Слева направо

Операторы

Выражение-оператор: `expression;`

Составной оператор: `{}`

Условный оператор: `if (expr) stmt; else stmt;`

`else` всегда относится к ближайшему `if`:

```
if (x > 2)
    if (y > z)
        y = z;
    else
        z = y;
```

```
if (x > 2) {
    if (y > z)
        y = z;
}
else
    z = y;
```

Операторы

Оператор выбора:

```
switch (expr) {
    case const-expr: stmt;
    case const-expr: stmt;
    default: stmt;
}
```

Оператор `break` — немедленный выход из `switch`.

Операторы

Цикл `while`: `while (expression) stmt;`

Цикл `do-while`: `do { stmt; } while (expression);`

Проверка условия выхода из цикла после выполнения тела

Цикл `for`:

```
for (decl1; expr2; expr3)
    stmt;
```

```
decl1;
while (expr2) {
    stmt;
    expr3;
}
```

`for ( ; ; ) stmt;` — бесконечный цикл

Операторы

Операторы `break` и `continue`: выход из внутреннего цикла и переход на следующую итерацию

Оператор `goto`: переход по метке

```
goto label;
...
label:
```

Областью видимости метки является *вся функция*