Курс «Алгоритмы и алгоритмические языки»
1 семестр 2015/2016

Лекция 6

Область действия переменной и классы памяти

```
#include <stdio.h>
int count;                      /* глобальная переменная */
void func (void)
{
  int count;                    /* автоматическая переменная */
  count = count - 2;
}
static int mult = 0;            /* статическая переменная */
int sum (int x, int y)
{
  static int first = 1;         /* статическая переменная */
  count++;
  return (x + y) * (++mult);
}
int main (void)
{
  register int s = 0;           /* регистровая переменная */
  count = 0;
  s += sum (5, 7);
  s += sum (9, 4);
  func ();
  printf ("Сумма равна %d, вызвали функцию %d раз\n", s, count);
  return 0;
}
```

Инициализация переменной

- При объявлении переменной:

```
int x = 42;
```

  - автоматические переменные инициализируются каждый раз при входе в соответствующий блок;

    если нет инициализации, значение соответствующей переменной не определено!
  - глобальные и статические инициализируются только

    один раз в начале работы программы;

    если нет инициализации, они обнуляются компилятором
  - внешние переменные инициализируются только в том файле, в котором они определяются
  - при инициализации переменной типа с квалификатором **`const`** она является константой и не может изменять свое значение

Литералы

- Литералы задают константу (фиксированное значение)
  - символьные константы **`'c'`**, **`L'%'`**, **`'\0x4f'`**, **`'\040'`**
    тип символьной константы – **`int`**!
  - целые константы **`100`**, **`-34l`**, **`1000U`**, **`999llu`**
  - константы с плавающей точкой **`11.123F`**, **`4.56e-4f`**, **`1.0`**, -**`11.123`**, **`3.1415926l`**, -**`6.626068e-34L`**
    тип вещественной константы без суффикса – **`double`**!
  - шестнадцатеричные константы **`0x80`** (*128*)
  - восьмеричные константы **`012`** (*10*)
  - строковые константы **`"a"`**, **`"Hello, World!"`**, **`L"Unicode string"`**
  - специальные символьные константы **`\n`**, **`\t`**, **`\b`**

Операции над целочисленными данными

- ***Арифметические***
  - *Одноместные*:
    изменение знака, или «одноместный минус» (**–**),
    одноместный плюс (**+**).
  - *Двухместные*:
    сложение (**+**), вычитание (**–**), умножение (*****),
    деление нацело (**/**), остаток от деления нацело (**%**).

```
(a/b) * b + (a%b) == a
```

- ***Отношения*** (результат 0/1 типа **`int`**)
  - больше (**>**), больше или равно (**>=**),
    меньше (**<**), меньше или равно (**<=**)
- ***Сравнения*** (результат 0/1 типа **`int`**)
  - равно **(==),** не равно **(!=)**
- ***Логические***
  - отрицание (**!**), конъюнкция **(&&),** дизъюнкция **(||)**
  - ложное значение – 0, истинное – любое ненулевое
  - **“ленивое” вычисление && и ||**

Операции присваивания

- ***Побочные эффекты:*** изменение объекта, вызов функции
- **`lvalue = rvalue`**
  - **`lvalue`** – выражение, указывающее на объект памяти
  - **`rvalue`** – выражение
  - Пример **`a = b = c = d = 0;`**
- ***Укороченное присваивание:*** **`lvalue op= rvalue`**
  где **`op -`** двухместная операция

  Пример **`a += 15;`**
- ***Инкремент и декремент*** (**`++`** и **`--`**)
  - префиксные и постфиксные
- ***Последовательное вычисление:*** операция запятая (**`,`**)

  Пример **`a = (b = 5, b + 2);`**

Точки следования

- ***Побочные эффекты:*** изменение объекта, вызов функции
- ***Точка следования*** **(*sequence point*)*:*** момент во время выполнения программы, в котором все побочные эффекты предыдущих вычислений закончены, а новых – не начаты
  - первый операнд **&&**, **||**, **,**
  - окончание ***полного*** выражения
  - между двумя точками следования изменение значения переменной возможно не более одного раза

    ```
    (a=2) + (a=3)
    ```

    ```
    i++ + ++i
    ```
  - старое значение переменной читается только для определения нового

    ```
    a = b++ + b
    ```

Форматный ввод-вывод

```
#include <stdio.h>

int main (void)
{
  int s = 0;
  int a, b;

  scanf ("%d%d", &a, &b);
  s += a + b;
  printf ("Сумма равна %d\n", s);
  return 0;
}
```

Форматный ввод-вывод

Спецификаторы ввода-вывода

%d, **%ld**, **%lld** – напечатать/считать число типа **int**, **long**, **long long**

%u, **%lu**, **%llu** – напечатать/считать число типа **unsigned**, **unsigned long**, **unsigned long long**

%f, **%Lf** – напечатать число типа **double**, **long double**

%f, **%lf** , **%Lf** – считать число типа **float** , **double**, **long double**

%c - напечатать/считать символ

%4d – вывести число типа **int** минимум в четыре символа

%.5f – вывести число типа **double** с пятью знаками

%% - напечатать знак процента

Функция **scanf** возвращает количество удачно считанных элементов

Пример Си-программы

```
/* Решение квадратного уравнения */
#include <stdio.h>
#include <math.h>
int main (void) {
  int a, b, c, d;
  /* Введем коэффициенты */
  if (scanf ("%d%d%d", &a, &b, &c) != 3) {
    printf ("Требуется ввести три коэффициента!");
    return 1;
  }
  if (!a) {
    printf ("Уравнение не квадратное!\n");
    return 1;
  }
  d = b*b – 4*a*c;
  if (d < 0)
    printf ("Решений нет\n");
  else if (d == 0) {
    double db = -b;
    printf ("Решение: %.4f\n", db/(2*a));
  } else {
    double db = -b;
    double dd = sqrt (d);
    printf ("Решение 1: %.4f, решение 2: %.4f\n", (db+dd)/(2*a), (db-dd)/(2*a));
  }
  return 0;
}
```

Пример Си-программы

```
/* Решение квадратного уравнения */
#include <stdio.h>
#include <math.h>
int main (void) {
  int a, b, c, d;
  /* Введем коэффициенты */
  if (scanf ("%d%d%d", &a, &b, &c) != 3) {
    printf("Требуется ввести три коэффициента!");
    return 1;
  }
  if (!a) {
    printf ("Уравнение не квадратное!\n");
    return 1;
  }
```

Пример Си-программы

```
  d = b*b – 4*a*c;
  if (d < 0)
    printf ("Решений нет\n");
  else if (d == 0) {
    double db = -b;
    printf ("Решение: %.4f\n", db/(2*a));
  } else {
    double db = -b;
    double dd = sqrt (d);
    printf ("Решение 1: %.4f, решение 2: %.4f\n",
            (db+dd)/(2*a), (db-dd)/(2*a));
  }
  return 0;
}
```