Курс «Алгоритмы и алгоритмические языки»
1 семестр 2018/2019

Лекция 7

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

```c
#include <stdio.h>
int main (void) {
  int s = 0;
  int a, b;
  scanf ("%d%d", &a, &b);
  s += a + b;
  printf ("Sum is %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

```c
/* 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) -> 255
    ```
  - Плавающий тип в целочисленный: отбрасывается дробная часть
  - “Широкий” плавающий тип в “узкий”: округление или усечение числа

- ***Явное приведение типов:*** **`(type) expression`**
  - Пример **`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

- ***Неявное приведение типов:*** происходит, когда операнды двухместной операции имеют разные типы
  - 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

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