Курс «Алгоритмы и алгоритмические языки»
1 семестр 2013/2014

Лекция 5

Типы данных

- Базовые типы данных: **`char`** (*символьный*), **`int`** (*целый*), **`float`** (*с плавающей точкой*), **`double`** (*двойной точности*), **`_Complex`** (C99, *комплексный*)
- Тип **`void`** (*без значения*)

- Модификаторы базовых типов: **`signed`**, **`unsigned`**, **`long`**, **`short`**, **`long long`** (C99)
  - к типу **`int`** применимы все модификаторы
  - к типу **`char`** – только **`signed`** и **`unsigned`**
  - к типу **`double`** – только **`long`** (C99)

Типы данных

- Представление целых чисел: позиционная двоичная система
  - Байты в представлении числа идут подряд
  - Порядок байт не гарантируется, то есть зависит от аппаратуры (big/little endian)
  - Порядок бит в байте также не гарантируется (и его может быть невозможно узнать)
  - Отрицательные числа часто представляются в дополнительном коде (*n* бит):

    - самый значащий бит (*n*-1) является знаковым

    - биты от 0 до *n*-2 – значения

    - положительные значения – как обычно

    - отрицательные значения: 2^n - |x|

Типы данных

- **`sizeof`** – размер типа (любого объекта типа)
  - **`int x -> sizeof(x) == sizeof(int)`**
  - Файл **`limits.h`** задает минимальные и максимальные значения целых типов
  - **`sizeof(char) == 1`**
    `∧`
    `sizeof(short)` ≥ **`2`**
    `∧`
    `sizeof(int)` ≥ **`2`**
    `∧`
    `sizeof(long)` ≥ **`4`**
    `∧`
    `sizeof(long long)` ≥ **`8`**
  - Файл **`inttypes.h`** задает знаковые и беззнаковые целые типы фиксированных размеров (8, 16, 32, 64 бита)

Типы данных

- Тип **`_Bool`** (C99, значения 0/1, целый беззнаковый)
  - Необходимо включить **`stdbool.h`** для объявлений **`bool`**, **`true`**, **`false`**
- Тип **`_Complex`** (C99, **`float/double/long double`**)
  - Необходимо включить **`complex.h`** для объявлений **`complex`**, **`I`** и т.п.
  - Тип **`_Imaginary`** (C99) является необязательным

Переменные

- Переменная = тип + имя + ***значение***

  Каждая переменная является *объектом* программы
- **`Ключевые слова`** (C89 – 32, C99 – C89 + 5) не могут быть именами переменных
- Объявление переменной:

  тип список_переменных

  Можно задать класс памяти и начальное значение переменной

Область действия переменной

- Переменная может быть объявлена:
  (1) внутри функции или блока (локальная);
  (2) в объявлении функции (параметр функции);
  (3) вне всех функций (глобальная).
- Область действия (видимости)
  - локальной переменной – блок, в котором она объявлена
  - глобальной переменной – программный файл, начиная со строки объявления
- В одной области действия нельзя объявлять более одной переменной с одним и тем же именем

Область действия переменной и классы памяти

```
#include <stdio.h>
int count;                      /* глобальная переменная */
static int mult = 0;            /* статическая переменная */
int sum (int x, int y)
{
  count++;
  return (x + y) * (++mult);
}
void func (void)
{
  int count;                    /* автоматическая переменная */
  count = count - 2;
}
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
    ```