Семинар 01.

Переменные, простейшие операции, ввод/вывод.

Содержание

- Понятие «переменная» с точки зрения любого языка программирования. Атрибуты переменной (имя, тип, адрес, значение, время жизни, область видимости).
- Требования к именам переменных в языке Си.
- Знакомство с типами int и float.
- Описание (в данном случае определение) переменных.
- Присваивание, простейшие арифметические операции.
- Структура простой программы, с учетом описания переменных.
- Вывод значение переменной (printf, строка форматирования, спецификаторы, esc-последовательности, ошибки при использовании printf).
- Ввод значения переменной (scanf, строка форматирования, алгоритм работы scanf, ошибки при использовании scanf).

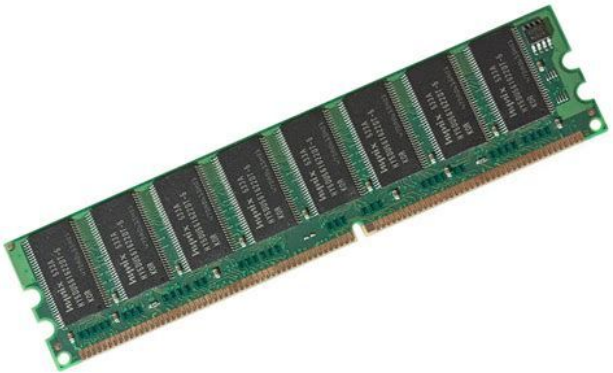
Стандарт C89/99/11

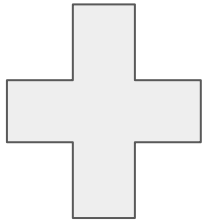
ANSI C — стандарт языка **C**, опубликованный Американским национальным институтом стандартов (ANSI). Следование этому стандарту помогает создавать легко портируемые программы.

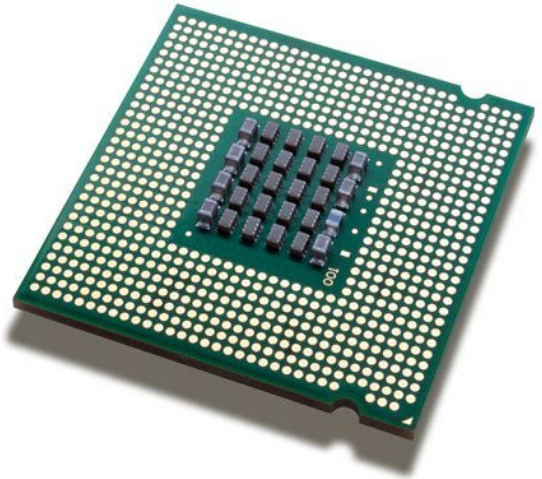
Любая программа, написанная только с использованием стандарта и не допускающая специфических аппаратных допущений, гарантированно должна работать на любой платформе с достаточно стандартной реализацией языка C.

http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf

Компьютер = Память + Процессор







Переменные

Переменная в программе представляет собой абстракцию ячейки памяти компьютера или совокупности таких ячеек.

Переменную можно охарактеризовать следующим набором атрибутов:

- имя (идентификатор);
- тип;
- адрес;
- значение;
- область видимости;
- время жизни.

Атрибуты переменной (1)

- *Имя* (идентификатор) – это строка символов, используемая для идентификации некоторой сущности в программе (переменной, функции и т. п.).

- *Тип* переменной определяет как переменная хранится, какие значения может принимать и какие операции могут быть выполнены над переменной.

- *Адрес* переменной – это ячейка памяти, с которой связана данная переменная.

Атрибуты переменной (2)

- *Значение* переменной – это содержимое ячейки или ячеек памяти, связанных с данной переменной.

- *Область видимости* переменной – это часть текста программы, в пределах которой переменная может быть использована.

- *Время жизни* переменной – это интервал времени выполнения программы, в течение которого переменная существует (т.е. ей выделена память).

Имена переменных (стр. 51, 404)

A.1.3 Identifiers

(6.4.2.1) *identifier:*

identifier-nondigit
identifier identifier-nondigit
identifier digit

(6.4.2.1) *identifier-nondigit:*

nondigit
universal-character-name
other implementation-defined characters

(6.4.2.1) *nondigit:* one of

```
_  a  b  c  d  e  f  g  h  i  j  k  l  m
   n  o  p  q  r  s  t  u  v  w  x  y  z
   A  B  C  D  E  F  G  H  I  J  K  L  M
   N  O  P  Q  R  S  T  U  V  W  X  Y  Z
```

(6.4.2.1) *digit:* one of

```
0  1  2  3  4  5  6  7  8  9
```

Ключевые слова (стр. 50)

6.4.1 Keywords

Syntax

keyword: one of

`auto`	`enum`	`restrict`	`unsigned`
`break`	`extern`	`return`	`void`
`case`	`float`	`short`	`volatile`
`char`	`for`	`signed`	`while`
`const`	`goto`	`sizeof`	`_Bool`
`continue`	`if`	`static`	`_Complex`
`default`	`inline`	`struct`	`_Imaginary`
`do`	`int`	`switch`	
`double`	`long`	`typedef`	
`else`	`register`	`union`	

Имена переменных (стр. 404)

```
 5     int a = 0;
 6     int _b = 0;
 7     int A = 0;
 8     int 1a = 0;
 9     int a1 = 0;
10     int for = 0;
```

```
$ gcc -o name.exe name.c
name.c: In function ‘main’:
name.c:8:6: error: invalid suffix "a" on integer constant
  int 1a = 0;
      ^~
name.c:8:6: error: expected identifier or ‘(’ before numeric constant
name.c:10:6: error: expected identifier or ‘(’ before ‘for’
  int for = 0;
      ^~~
```

Типы переменных. Integer (стр. 38, 6.2.6.2)

Для беззнаковых целых (unsighned integer) каждый из **N** бит в переменной является степенью **2** от 1 до **2^(N-1)**.

Для знаковых целых (signed integer) добавляется бит знака. Для отрицательных значений он равен **1.**

Диапазон значений от **-2^N** до **2^N**

Типы переменных. float

Переменные типа float могут хранить вещественные числа, гораздо большего значения чем числа, описываемые типом int. Кроме того, они могут хранить числа с дробной частью (-0.723, 265.12). НО арифметические операции для таких переменных выполняются медленнее, а сами числа представлены приближенно.

Определение переменных

До того, как переменная будет использована в программе, она должна быть *определена*. Чтобы определить переменную, необходимо сначала указать ее тип, а затем имя.

```
 5    int i;
 6    float a, b;
 7    int j = 0;
 8    float c = 3.14;
 9    float i;
10    float a;
```

Определение переменных

```
sem01.c: In function ‘main’:
sem01.c:9:8: error: conflicting types for ‘i’
  float i;
        ^
sem01.c:5:6: note: previous declaration of ‘i’ was here
  int i;
      ^
sem01.c:10:8: error: redeclaration of ‘a’ with no linkage
  float a;
        ^
sem01.c:6:8: note: previous declaration of ‘a’ was here
  float a, b;
        ^
```

Структура программы

- Стандарт C89 требует, чтобы определения переменных располагались в начале функций.

- Стандарт C99 снимет это ограничение и позволяет смешивать определения переменных и операторов. Единственное ограничение – переменная должна быть описана перед своим использованием.

```c
#include <stdio.h>

int main(void)
{
    //определения
    int i = 0;

    //операторы
    return 0;
}
```

printf (printf formatted) [стр. 274, #7.19.6, man fprintf]

```c
#include <stdio.h>

int main(void)
{
    int i = 2018;
    printf("Hello world\n");
    printf("Hello world, %d\n", i);
    printf("Hello world, %f %f", 3.14, 2.72);
}
```

```
Hello world
Hello world, 2018
Hello world, 3.140000
```

Printf. Ошибки.

```c
#include <stdio.h>

int main(void)
{
    int i = 2018;
    printf("Hello world\n", i);
    printf("Hello world, %d\n", i, i);
    printf("Hello world, %d %d\n", i);
    printf("Hello world, %d %f", 3.14, i);
}
```

printf. Ошибки.

```
└─>$ cc -o sem04 sem04.c
sem04.c: In function ‘main’:
sem04.c:6:9: warning: too many arguments for format [-Wformat-extra-args]
  printf("Hello world\n", i);
         ^
sem04.c:7:9: warning: too many arguments for format [-Wformat-extra-args]
  printf("Hello world, %d\n", i, i);
         ^
sem04.c:8:9: warning: format ‘%d’ expects a matching ‘int’ argument [-Wformat=]
  printf("Hello world, %d %d\n", i);
         ^
sem04.c:9:9: warning: format ‘%d’ expects argument of type ‘int’, but argument 2 has type ‘double’ [-Wformat=]
  printf("Hello world, %d %f", 3.14, i);
         ^
sem04.c:9:9: warning: format ‘%f’ expects argument of type ‘double’, but argument 3 has type ‘int’ [-Wformat=]
```

```
$ ./printf.exe
Hello world
Hello world, 2018
Hello world, 2018 -14084
Hello world, 1374389535 0.000000
```

Спецификаторы

В строке форматирования спецификатор обозначает место, в которое будет выведено соответствующее значение во время отображения строки.

Спецификаторы начинаются с символа “%”, а заканчиваются «символом преобразований», который определяет значение какого типа будет отображаться на экране.

Спецификатор	**Отображение аргумента**
%d	Целое число (100)
%f	Вещественное число (7.123000)
%e	Вещественной число в экспоненциальной форме (7.123000e+000)
%g	Что короче из %f или %e (7.123).

- **`%[ширина][.точность]тип`**
- **`%-[ширина][.точность]тип`**

Спецификаторы. Пример.

```c
#include <stdio.h>

int main(void)
{
    int i = 2018;
    float f = 3.1415;
    printf("Hello world, |%d|%8d|%-8d|%8.5d|%-8.3d|\n", i, i, i, i, i);
    printf("Hello world, |%f|%8f|%-8f|%8.5f|%-8.3f|\n", f, f, f, f, f);
}
```

scanf (scan formatted) [стр. 274, #7.19.6, man fscanf]

Алгоритм работы

1. Работа функции scanf управляется строкой форматирования.
2. Для каждого спецификатора scanf пытается выделить данные соответствующего типа во входных данных. Функция останавливается на символе, который не относится к очередному вводимому значению.
3. Если значение успешно прочитано, scanf продолжает обрабатывать строку. В противном случае - прекращает работу.

scanf. Пример.

```c
#include <stdio.h>

int main(void)
{
    int i = 2018;
    float f = 3.1415;
    scanf("%d %f", &i, &f);
    printf("Hello world, %d %f", i, f);
}
```

scanf. Пример. Вывод.

```
└─>$ ./sem06
1 2
Hello world, 1 2.000000
```

```
└─>$ ./sem06
1
2
Hello world, 1 2.000000
```

```
└─>$ ./sem06
1 hello
Hello world, 1 3.141500
```

```
└─>$ ./sem06
hello    1  2.72
Hello world, 1 2.720000
```

```
└─>$ ./sem06
1 2.7e1
Hello world, 1 27.000000
```

```
└─>$ ./sem06
2.3 4
Hello world, 2 0.300000
```

```
└─>$ ./sem06
1 2 3
Hello world, 1 2.000000
```

```
└─>$ ./sem06
hello 2.72
Hello world, 2018 3.141500
```

Строка форматирования scanf

```c
#include <stdio.h>

int main(void)
{
    int i = 2018;
    float f = 3.1415;
    scanf("%d, %f", &i, &f);
    printf("Hello world, %d %f", i, f);
}
```

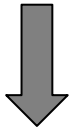
```
└─>$ ./sem06
1 2.72
Hello world, 1 3.141500
```

```
└─>$ ./sem06
1   ,   2.72
Hello world, 1 3.141500
```

```
└─>$ ./sem06
1, 2.72
Hello world, 1 2.720000
```

Ошибки scanf

1. Те же что у printf
2.
```
7     scanf("%d, %f", i, f);
```



```
└─>$ cc -o sem06 sem06.c
sem06.c: In function ‘main’:
sem06.c:7:8: warning: format ‘%d’ expects argument of type ‘int *’, but argument 2 has type ‘int’ [-Wformat=]
  scanf("%d, %f", i, f);
        ^
sem06.c:7:8: warning: format ‘%f’ expects argument of type ‘float *’, but argument 3 has type ‘double’ [-Wformat=]
```

Пример.

```c
#include <stdio.h>

int main(void)
{
    int rub = 0;
    float rate, btc = 0;
    printf("Enter number of RUB and exchange rate:");

    //Ex: 1000 0.000002
    scanf("%d %f", &rub, &rate);
    btc = rub * rate;
    printf("You have, %f BTC", btc);
    return 0;
}
```

```
└─>$ ./sem07
Enter number of RUB and exchange rate:100 0.000002
You have, 0.000200 BTC
```

Итого.

- Лабораторная №1
- Повторить семинар
- Читать стандарт http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1256.pdf
- Установить MSYS2
- Вопросы: @qwer_ty