Тема : знакомство с C (Си)

Содержание: введение структура программы декларирование переменных ввод/вывод

1. Введение

- **Необходимо создать каталог, в котором будут храниться написанные программы.**

mkdir <имя каталога>

- При написании программ можно использовать текстовый редактор **Scite** или любой другой текстовый редактор, имеющийся в системе.
Для запуска редактора записываем в **Terminal**-е команду:

scite &

После запуска **Scite** произвести следующие установки:

- *Languages -> C/C++*
- *View -> Line Numbers*

NB! Программу время от времени необходимо сохранять, в том числе и перед компилированием. **Обратите внимание, что необходимо указывать расширение файла.** В общем случае используется расширение ***.c** (например, **hello.c**).

- Программу компилируем командой:

```
gcc -o hello hello.c
```
или
```
gcc hello.c -o hello
```

gcc – С компилятор
-o – ключ, который позволяет назначить определенное имя исполнимому файлу
hello – здесь имя исполнимого файла
hello.c – файл, содержащий исходный текст программы

Можно компилировать и так:

```
gcc hello.c
```

в этом случае исполнимый файл будет называться **a.out**

- При компилировании возможны следующие ситуации:
  - **Никаких сообщений нет, на terminal-е появилась только еще одна пустая командная строка.** Такая ситуация наиболее хорошая и означает, что в тексте программы ошибки не найдены, программу можно запускать. Хотя это и не означает, что при составлении программы не были допущены логические ошибки. Логические ошибки проявляются во время работы программы.
  - **Появляется список ошибок вместе с номерами строк, где эти ошибки находятся.**

    В этом случае ошибки необходимо исправить:

    - Исправление ошибок желательно начать с первой по порядку;
    - Внимательно прочитать описание ошибки на английском языке. В общем случае эти пояснения позволяют точно определить причину ошибки.

    После того как исправления внесены в текст программы, надо сохранить программу и снова откомпилироват .

  - **Появляются предупреждения (warning).** Возможны ситуации, когда их возможно проигнорировать. Однако в некоторых ситуациях предупреждения означают, что в программе есть ошибки, которые не препятствуют выполнению программы, но с большой вероятностью программа может работать неверно. Поэтому всегда стоит с большим вниманием читать содержание warning-а.

- При работе на рабочих станциях **Linux** и использовании **gcc** могут понадобиться дополнительные ключи.

Например

В случае использования библиотеки **математических** функций

```
#include <math.h>
```

компилирование производим командой

```
gcc -o hello hello.c -lm
```

Info по различным функциям языка С можно получить следующим образом:

man ***имя_функции***

Например:

man sin

Выводится info о функции (в данном случае **sin**; часть SYNOPSIS выдает необходимые для компилирования дополнительные ключи):

*(**NB!** здесь приведена только часть информации)*

```
NAME
     sin - sine function

SYNOPSIS
     cc [ flag ... ] file ... -lm [ library ... ]
     #include <math.h>

     double sin(double x);

DESCRIPTION
     The sin() function computes the  sine  of  its  argument  x,
     measured in radians.

RETURN VALUES
     Upon successful completion, sin() returns the sine of x.

     If x is NaN or +Inf, NaN is returned.
```

Запуск программы с командной строки **Terminal**-а:

```
./hello
```

или

```
./a.out
```

2. Структура программы

Рассмотрим программу на С.

Упражнение 1:

Код приведенной ниже программы воспроизвести в текстовом редакторе, исследовать текст программы, откомпилировать и запустить на выполнение.

```c
/* первая программа на языке C*/

#include <stdio.h>

int main(void)
{
   printf("\nMulle meeldib progeda!\n");

//    getchar();
//    getchar();
// можно использовать, чтобы предотвратить
// закрытие окна в некоторых системах
   return 0;

}
```

пояснения

```c
/* первая программа на языке C – Стиль C */

//первая программа на языке C – Стиль C++
```

Это комментарии. Комментарии не мешают выполнению программы. Строка, начинающаяся с `//` и до своего конца, является комментарием. Многострочные комментарии используют знаки `/*` (для определения начала комментария) и `*/` (для конца).

```c
#include <stdio.h>
```

Здесь записываются команды препроцессора. В текст программы добавляют описания (прототипы) стандартных функций, содержащихся в библиотеках.

```c
int main(void)
```

Главная программа – функция с именем **main**. Именно с этой функции начинается выполнение С-программы вне зависимости от нахождения функции **main** в тексте программы. Возвращаемый код ошибки представляется целым числом **(int).** Данное в скобках ключевое слово **void** показывает, что в данном случае у **main** нет параметров. (Естественно, что при необходимости для функции **main** можно задавать параметры.)

```c
{}
```

Между этими фигурными скобками располагается описание функции **main**. Описание всех С-функций и в общем случае располагается между **{}.** Скобке «{» можно придать значение **«начало»,** а скобке «}» **- «конец».**

В больших программах обязательно присутствует декларация используемых переменных.

```
printf("\nMulle meeldib progeda!\n");
```

printf - это С-функция вывода на экран. В данном случае выводится **Mulle meeldib progeda!**, после чего производится переход с текущей строки на следующую (**\n**).

```
return 0;
```

В конце программа возвращает операционной системе целое значение 0. Это показывает, что программа успешно завершилась (без ошибок - это и означает код ошибки 0).

Итак:
Общая структура написанной на языке С программы следующая:

- предписания препроцессора
- прототипы функций (определение типа параметров функции и возвращаемого значения)
- переменные (глобальные и локальные)
- функции
- NB! Функция **main** в каждой программе обязательна.

3. Декларирование переменных

Пусть необходимо произвести некоторые вычисления. Пусть имеются два значения: 5 и 2. Затем к первому надо прибавить 1, а результат запомнить. Таким образом, у нас в памяти будет 2 значения: 6 и 2. Затем, предположим, из первого значения надо вычесть второе – получим 4 (значение 6 мы хотим сохранить).

Описанные действия на С записываются следующим образом:

```
a = 5;
b = 2;
a = a + 1; /* a=6 */
tulemus = a-b;
```

Обратите внимание, что необходимо в конце каждого действия ставить «;»

Данная программа может выглядеть следующим образом:

Упражнение 2:

Код приведенной ниже программы воспроизвести в текстовом редакторе, исследовать текст программы, откомпилировать и запустить на выполнение.

```c
#include <stdio.h>

int main(void)
{
//декларирование переменных
int a, b, tulemus;

//вычисления и вывод на экран
a = 5;
b = 2;
printf("\nEsimene arv on: %d\n",a);
printf("\nTeine arv on: %d\n",b);

printf("\nNüüd teeme nii!\n  Esimesele arvule
liita üks - %d+1: \n",a);

a = a + 1;
printf("\nPraegu esimene arv on: %d\n",a);

tulemus = a-b;

printf("\n %d - %d = %d\n",a,b,tulemus);

return 0;

}
```

Типы переменных в языке С:

Тип	Декларирование имени типа
Целочисленный	`int, long, short`
Переменная, состоящая из одного символа	`char`
Переменная, состоящая из ряда символов, символьная строка	`char[`*`число символов+1`*`]`
Плавающие типы (с плавающей точкой)	`float, double`

- Декларирование целых чисел (например, номер года, возраст и т.д.):

```
int yksTaisArv;
short teineTaisArv;
long kolmasTaisArv;
```

- Символьная переменная (например, просто символ алфавита):

```
char yksSymbol;
```

- Переменная, состоящая из некоторого количества символов, т.е. символьная (текстовая) строка (например, имя или фамилия):

```
char symboliteJada[21];
```

NB! При декларирование символьной строки необходимо к желаемому количеству символов прибавить еще один, так как концом текстовой строки в С считается нулевой байт **\0.**

- Плавающие типы:

```
float yksMurdArv;
double teineMurdArv;
```

Декларирование констант

С помощью ключевого слова **const** в программе можно использовать переменные с фиксированным значением, которые изменить нельзя. Ключевое слово const введено стандартом **ANSI**. Квалификатор const применяется, чтобы разместить объекты в памяти, открытой только на чтение (Стандарт заимствует из **C++**).

В С чаще используется деректива препроцессора **#define**.

Пример:

Упражнение 3:

Код приведенной ниже программы воспроизвести в текстовом редакторе, исследовать текст программы, откомпилировать и запустить на выполнение.

```
#include <stdio.h>
//#define jooksevAasta 2010

int main(void)
{
// декларирование переменных
const int jooksevAasta = 2010;
int vanus, synniAasta;
```

```
// вычисления и вывод на экран
synniAasta = 1988;
vanus = jooksevAasta - synniAasta;

printf("\nVanus on %d aastat\n",vanus);

return 0;
}
```

4. Ввод/вывод

Примечание: приведенная здесь информация — лишь малая часть функций ввода/вывода языка С. Полный обзор функций ввода/вывода можно найти, например, здесь:

- http://www.gnu.org/software/libc/manual/html_node/ *: **I/O on Streams***

Там же можно получить более детальную информацию по GNU C функциям ввода/вывода в следующих разделах:

- 11 Input/Output Overview
- 12.7 Simple Output by Characters or Lines
- 12.8 Character Input
- 12.9 Line-Oriented Input
- 12.14.8 Formatted Input Functions
- 12.14.8 Formatted Input Functions

Далее рассмотрим:

Ввод:

- **Форматированный ввод - функция scanf**
- **Неформатированный ввод - функции getchar(), gets()**

Вывод:

- **Форматированный вывод - функция printf**
- **Неформатированный вывод - функции putc(), puts()**

Пример

- Пример программы ввода/вывода

Упражнение 5 (работа выполняется в классе)

- Упражнение по этой теме находится в конце данного документа

ФОРМАТИРОВАННЫЙ ВВОД - функция scanf

Прототип функции находится в файле **stdio.h**

scanf ("строка форматов", &переменная1, &переменная2, ...)

Параметрами функции **scanf** являются строка форматов (сначала следует % и символ формата, соответствующий типу вводимой переменной) и перечень адресов вводимых переменных.

Примеры использования:

- Ввод целых чисел:

```
scanf("%d",&taisArv);

/*
d — показывает, что вводимая переменная – целочисленного типа (int, short, long);


В строке форматирования могут использоваться префиксы. Например,
в случае переменных целочисленного типа можем использовать  %ld для long int;

taisArv — это имя вводимой переменной;
Находящийся перед именем переменной амперсанд & обязателен, это адрес вводимой переменной taisArv.
*/
```

- Ввод переменных плавающих типов:

```
scanf("%lf",&murdArv);
/*
f — показывает, что вводимая переменная – число плавающего типа (float,double);
в случае чисел плавающего типа float – форматирование %f;
в случае чисел плавающего типа double – форматирование %lf;
```

```
murdArv — это имя вводимой переменной;
Находящийся перед именем переменной амперсанд & обязателен, это адрес
вводимой переменной murdArv
*/
```

- Ввод текстовых переменных:

a) Ввод переменной, состоящей из одного символа:

```
scanf("%c",&yksSymbol);

/*
c — показывает, что вводимая переменная – символ (char);

yksSymbol — это имя вводимой переменной;
Находящийся перед именем переменной амперсанд & обязателен, это адрес
вводимой переменной Находящийся перед именем переменной амперсанд &
обязателен, это адрес вводимой переменной yksSymbol
*/
```

b) Ввод переменной, состоящей из ряда символов:

```
scanf("%s",pikkSona);
/*
s — показывает, что вводимая переменная – строка символов (char[число
символов – 1]);

pikkSona — это имя вводимой переменной;

NB! в этом случае не надо использовать символ &, т.к. переменная сама
является адресом
*/
```

НЕФОРМАТИРОВАННЫЙ ВВОД – функция `getchar()`

Функция для ввода одного символа:

```
getchar(yksSymbol);
```

НЕФОРМАТИРОВАННЫЙ ВВОД – функция `gets()`

Функция для ввода ряда символа (*get string*):

```
gets (symboliJada);
```

NB! При использовании функции gets() надо быть очень внимательным, т.к. нет ограничения ряда вводимых символов, необходимо учитывать, что чтение производится до тех пор, пока не встретится перевод строки или конец файла.

ФОРМАТИРОВАННЫЙ ВЫВОД — функция printf

Прототип функции находится в файле **stdio.h**

```
printf("строка форматов", переменная1, переменная2, ...)
```

Параметрами функции **printf** являются строка форматов (сначала следует % и символ формата, соответствующий типу вводимой переменной) и перечень выводимых переменных.

Примеры использования:

- Вывод целых чисел:

```
printf("Täisarv on: %d",taisArv);
/*
d — показывает, что выводимая переменная - целочисленного типа (int, short, long);

в случае переменных целочисленного типа long можем использовать форматирование %ld;
taisArv — это имя выводимой переменной;
*/
```

- Вывод переменных плавающих типов:

```
printf("Murdarv on %f",murdArv);
/*
f — это строка показывает, что выводимая переменная - число плавающего типа (float,double);

в случае чисел плавающего типа float используем форматирование %f;
long double - %lf или %Lf, для double lL игнорируется;
murdArv — это имя выводимой переменной;
*/
```

Пример

```
printf("Vormindatud murdarv on %-7.2f",murdArv);
```

```
/*
f — показывает, что выводимая переменная - число плавающего типа (float,double);
в случае чисел плавающего типа float используем форматирование %f;
long double - %lf или %Lf, для double lL игнорируется;
murdArv — это имя выводимой переменной;

%-7.2f означает:

    - выравнивание по левому краю
    7 показывает количество позиций, которое переменная занимает на экране
    2 количество позиций после запятой
*/
```

- Вывод символьных переменных

  a) Вывод переменной, состоящей из одного символа:

```
printf("Üks sümbol on %c",yksSymbol);

/*
c —показывает, что выводимая переменная – символ (char);
yksSymbol — это имя выводимой переменной
*/
```

  b) Вывод переменной, состоящей из ряда символов:

```
printf("Mitmest sümbolist koosnev
märkmuutuja on %s",pikkSona);

/*
s — показывает, что выводимая переменная – строка символов
(char[число символов – 1]);
pikkSona – это имя выводимой переменной;
*/
```

В одной команде можно выводить несколько переменных (здесь **printf** записывается в одну строку):

```
printf ("Täisarv on %d, murdarv on %f, symbol on
%c, sõna on %s", taisArv, murdArv, yksSymbol,
pikkSona);
```

НЕФОРМАТИРОВАННЫЙ ВЫВОД – функция putc()

Функция для вывода одного символа:

```
putc(yksSymbol);
```

НЕФОРМАТИРОВАННЫЙ ВЫВОД – функция puts()

Функция для вывода ряда символа (*put string*), заменяет ('\0') символом конца строки ('\n'):

```
puts(symboliteJada);
```

Пример использования:

```
puts("See on üks rida");
```

Упражнение 4:

Код приведенной ниже программы воспроизвести в текстовом редакторе, исследовать текст программы, откомпилировать и запустить на выполнение.

```
#include <stdio.h>

/* Программа приветствует пользователя и
спрашивает имя и год его рождения. Считает
возраст пользователя, выводит возраст на экран
вместе с именем пользователя. */

int main(void)
{
/*
декларирование переменных:
nimi — имя пользователя, которое вводит
пользователь, длиной 20 символов;
synniAasta — год рождения пользователя, вводит
пользователь,целое число;
sinuVanus — возраст пользователя, вычисляет
программа, целое число
*/
    char sinuNimi[21];
    int synniAasta, sinuVanus;
    int jooksevAasta = 2010;

//Выводится приветствие и спрашивается имя
//пользователя
printf("Tere!\n");
printf("Mina olen arvuti. Mis on sinu nimi?\n");

//пользователь вводит имя: набирает на
клавиатуре свое имя и нажимает
//клавишу ENTER; переменная sinuNimi получает
//в качестве своего значения
//строку введенных символов
  scanf("%s", sinuNimi);
//спрашивается год рождения пользователя
  printf("Tore! Mis aastal syndisid?\n");
```

```
// пользователь вводит год рождения:
// набирает на  клавиатуре свой
// год рождения и нажимает клавишу ENTER;
   переменная synniAasta
// получает в качестве своего
// значения введенное число
   scanf("%d",&synniAasta);

//расчет возраста

sinuVanus= jooksevAasta-synniAasta;

//вывод результата;
//выводятся строковая
// переменная и целое число
printf("\nTere %s, sa oled %d aastat vana\n",
sinuNimi, sinuVanus);

//    getchar();
//    getchar();
    return 0;
}
```

При запуске данная программа выдает следующий диалог (**жирный шрифт — спрашивает компьютер,** *курсив — вводит пользователь*)

```
Tere, mina olen Arvuti! Mis on sinu nimi?
Juri
Tore! Mis aastal sa syndisid?
1990
Tere, Juri, sa oled 20 aastat vana.
```

Упражнение 5:

Составить программу, которая при запуске выдает следующий диалог с пользователем. (**жирный шрифт – выводит компьютер,** *курсив – вводит пользователь*):

Teeme nyyd veidi matemaatikat. (Посчитаем!)
Sisesta kaks arvu, mida tahad liita. (Введи 2 числа, которые хочешь сложить)

Sisesta esimene arv (Введи первое число): *13*
Sisesta teine arv (Введи второе число): *23*
Arvude 13 ja 23 summa on 36 (Сумма чисел 13 и 23 = 36).

Siseta kaks arvu, mida tahad korrutada. (Введи 2 числа, которые хочешь умножить)
Sisesta esimene arv (Введи первое число): *5*
Sisesta teine arv (Введи второе число): *3*
Arvude 5 ja 3 korrutis on 15 (результат умножения чисел 5 и 3 = 15).

Nyyd jagame! (Теперь делим!)
Sisesta jagatav (Введи делимое): *17*
Sisesta jagaja (Введи делитель): *6*
Kui jagame 17.00 arvuga 6.00 siis on tulemus 2.83 (Если делим 17.00 на 6.00 получаем 2.83).

Trigonomeetria! (Тригонометрия!)
Sisesta arv x (radiaanides), ma arvutan sulle sin(x) (Ввести число x (в радианах), я тебе посчитаю sin(x)): *4*
Sin (4.00) on -0.75680 (Sin (4.00) равняется -0.75680)

Указания:

- Все используемые числа - переменные
  - при сложении и умножении можно использовать как и целочисленные типы (**int**, **short** или **long**), так и плавающие (**float, double**)
  - при делении использовать плавающие типы (**float, double**)
  - при вычислении синуса использовать плавающие типы (**float, double**)
- продекларировать нужно все используемые переменные:
  - со стороны пользователя вводимые переменные
  - используемые программой для хранения результатов переменные (например, для хранения суммы и т.д.)
- переход на следующую строку получаем при помощи **\n**
- вычисления производятся только с введенными пользователем числами

➢ Основные арифметические операции, используемые в языке С:

«+» — сложение
«-» — вычитание
«*» — умножение
«/» — деление

Например,

```
int a;
int b;
int c;
...
c=a-b;
```

- лучше выбирать смысловые имена переменных
- Т.к. для использования функции вычисления синуса **sin** необходимо использовать библиотеку **math.h**, то добавим в начало кода программы строку

```
#include <math.h>
```

Прототип функции **double sin(double x)** (см. описание функции) означает, что функция **sin** принимает параметр типа **double** и возвращает тип **double.** Если мы хотим разместить результат в переменной **tulemus**, то должны записать

tulemus=sin(a);

Здесь тип переменной **a** — **double.**

Отчет оформляется в MS Word (или в другом текстовом редакторе), представляется в распечатанном виде. Отчет должен содержать необходимое для понимания описание работы. Представляемая программа должна быть прокомментирована.
Использованы материалы:

- ***Керниган Б., Ритчи Д. Язык С.*** (K&R)
- ***Teodor Luczkowski****. Baasteadmised programmeerimiskeelest C++*
- ***A. D. Marshall.*** *Programming in C. UNIX System Calls and Subroutines using C.* *http://www.cs.cf.ac.uk/Dave/C/CE.html [18.10.2010]*
- ***The cplusplus.com tutorial.*** *Complete C++ language tutorial* *http://www.cplusplus.com/doc/tutorial/index.html [18.10.2010]*
- ***The GNU C library*** *http://www.gnu.org/software/libc/manual/html_node/ [18.10.2010]*
- ***Материалы*** *Helena Kruus (MSc)*

Марина Брик

Составлено: 23.10.2007
Обновлено: 30.09.2008
Обновлено: 18.10.2010