Тема : знакомство с C (Си)

параметры main, IF-ELSE, упражнения

Содержание: параметры (аргументы) функции main необходимые функции преобразования упражнение 1 2 IF-ELSE упражнение 3

- ### Параметры (аргументы) функции `main`

На предыдущем уроке познакомились с самой важной функцией языка **C** - функцией **main**. Напомним, что первая программа выглядела так:

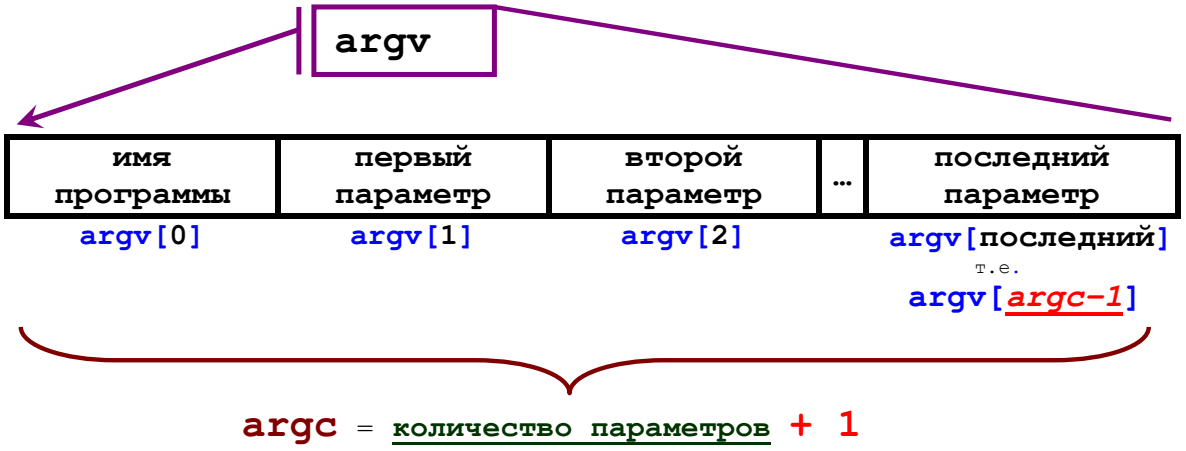
```
/* первая программа на языке C*/

#include <stdio.h>

int main(void)
{
   printf("\nMulle meeldib progeda!\n");
   return 0;
}
```

Слово в скобках **void** после имени функции **main** означает, что у данной функции нет параметров (аргументов). Однако, может возникнуть такая ситуация, что надо передать функции **main** значения некоторых параметров. Что тогда делать?

Для того чтобы функция **main** была бы способна использовать заданные параметры, необходимо записать следующее**:**

```
int main(int argc, char *argv[])
```

На первом месте параметр **argc** – целое число, чье значение на единицу больше количества параметров, передаваемых программе (таким образом, если программе передаются 3 параметра, то значением **argc** будет 4), **argv** – массив указателей (массив - множество некоторых данных, о массивах поговорим позже) на имя программы и передаваемые параметры. Параметры передаются программе как строковые переменные.



Рассмотрим программу, знакомую нам из Занятия 1(уже модифицированную):

Упражнение 1:

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

```
/*
ИМЯ файла:progc.c
   АВТОР:
   ДАТА создания:
   ДАТА изменения:
   ОПИСАНИЕ:Программа считывает с командной строки имя пользователя и год рождения
         и выдает данные на экран
*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
//      декларирование переменных:
    char sinuNimi[21];
    int synniAasta;

printf("Mina olen arvuti.\n");
printf("Selle programmi nimi on %s\n",argv[0]);

//копирование параметров в соответствующие локальные переменные
strcpy(sinuNimi,argv[1]);
synniAasta=atoi(argv[2]);

//вывод результата; выводятся строковая переменная и целое число
printf("\nSinu nimi on %s aastat vana\n  Sa syndisid %d aastal", sinuNimi, synniAasta);

    return 0;
}
```

Для компиляции этой программы на **Terminal**-е запишем

```
gcc -o progc progc.c
```

Если откомпилируем программу успешно без ошибок, запустим ее следующим образом

```
./progc Juri 1989
```

Различие между программой, составленной ранее, и данной состоит в том, что ранее ввод значений переменных (имя пользователя и его год рождения) осуществлялся в ходе работы программы, теперь же соответствующие значения задаются при запуске программы.

Вывод программы будет следующий:

```
Tere!
Mina olen arvuti.
Selle programmi nimi on ./progc

Sinu nimi on Juri.
  Sa syndisid 1989 aastal.
```

Так как при таком использовании программа получает параметры в виде строк, то необходимо произвести преобразования в соответствующие нуждам типы переменных: целочисленные (**int, short, long**), с плавающей запятой (**float, doubl**e) или в строковые переменные фиксированной длины (**char[***elementide_arv+1***]**). В приведенном примере сделаны преобразования в строковые переменные фиксированной длины и в целые числа.

```
//копирование параметров в соответствующие локальные переменные
strcpy(sinuNimi,argv[1]);
synniAasta=atoi(argv[2]);
```

Напомним, что **argv[0]** содержит имя программы, оставшийся ряд параметров (начиная с **argv[1]**)) – заданные при запуске программы параметры. Здесь: **argv[1] – Juri, argv[2] – 1989.**

С помощью функции **strcpy()** помещаем заданный параметр **Juri** в строковый массив (в строковую переменную) **sinuNimi.** С помощью функции **atoi()** (*argument to integer*)целочисленная переменная **synniAasta** преобретает значение последнего параметра **1989.**

Необходимые функции преобразования

- **Из строки в заданные типы:**

Описания соответствующих функций находится в библиотеке <stdlib.h>, в соответствии с этим в начало программы помещаем:

```
#include <stdlib.h>
```

Функции преобразования:

- Строка (**char[]**) в целое число (**int**):
  `taisArv=atoi(string);`
- Строка (**char[]**)в целое число (**long**):
  `TaisArv=atol(string);`
- Строка (**char[]**) в тип с плавающей запятой (**double**):
  `reaalArv=atof(string);`
- **Заданные типы в строковые переменные:**

Описания соответствующих функций находится в библиотеке **<stdlib.h>**, в соответствии с этим в начало программы помещаем:

```
#include <stdlib.h>
```

Функции преобразования:

- целое число (**int**) в строку (**char[]**):
  `string=itoa(taisarv);`

- целое число (l**ong**) в строку (**char[]**):
  `string=ltoa(taisarv);`

- тип с плавающей запятой (**double**) в строку (**char[]**):
  `string=ftoa(reaalarv);`

- **Для строковой переменной:**

Описание соответствующей функции находится в библиотеке **<string.h> >**, в соответствии с этим в начало программы помещаем:
`#include <string.h>`

Функции преобразования:

`strcpy(nimi,parameeter);`

Упражнение 2:

Открыть созданную на предыдущем уроке программу (см. упражнение **Занятия 1** по С). Используя ее и выше приведенный пример, создать 2 состоящие только из функции **main** программы.

программа 1

- Запустить с командной строки:
  `./progc1 Juri 15 7`

  (в качестве параметров задается имя и 2 целочисленных параметра)

- В ходе программы выдается на экран:

```
Tere, Juri!
Liidame kaks arvu kokku.
Esimene liidetav on 15
Teine liidetav on 7
Arvude 15 ja 7 summa on 21
```

- **NB!** Программа должна преобразовать вводимые данные в целочисленные(**`int`**, для этого нужна функция **`atoi`**)

NB! Сумму переменных должна вычислить сама программа (**NB!** Декларирование переменных!)

программа 2

- Запустить с командной строки:
  `./progc2 Heli 15 7`

  (в качестве параметров задается имя и 2 параметра с плавающей запятой)

- В ходе программы выдается на экран:

```
Tere, Heli!
Proovime nüüd jagamist.
Jagatav on 15.00
Jagaja on 7.00
```

```
Kui jagame 15.00 arvuga  7.00, on tulemuseks   2.143
```

- **NB!** Программа должна преобразовать вводимые данные в данные с плавающей запятой (**double**, для этого нужна функция **atof**)

NB! Частное должна вычислить сама программа (**NB!** Декларирование переменных!)

• Оператор IF - ELSE

Оператор **if - else** используется при необходимости сделать выбор. Формально синтаксис имеет вид:

```
if   (выражение)
        оператор-1
else
        оператор-2
```

Где часть **else** является необязательной. Сначала вычисляется выражение; если оно "истинно" (т.е. значение выражения отлично от нуля), то выполняется оператор-1. Если оно ложно (значение выражения равно нулю), и если есть часть с **else**, то вместо оператора-1 выполняется оператор-2. Так как **if** просто проверяет численное значение выражения, то возможно некоторое сокращение записи. Самой очевидной возможностью является запись

```
if   (выражение)
```

вместо

```
if   (выражение !=0)
```

иногда такая запись является ясной и естественной, но временами она становится загадочной. То, что часть **else** в конструкции **if** - **else** является необязательной, приводит к двусмысленности в случае, когда **else** опускается во вложенной последовательности операторов **if**. Эта неоднозначность разрешается обычным образом - **else** связывается с ближайшим предыдущим **if**, не содержащим **else**.

Например, в

```
if (N > 0)
   if (A > B)
           Z = A;
   else
           Z = B;
```

конструкция **else** относится к внутреннему **if**, как мы и показали, сдвинув **else** под соответствующий **if**. Если это не то, что вы хотите, то для получения нужного соответствия необходимо использовать фигурные скобки:

```
if (N > 0)
        {
        if (A > B)
        Z = A;
          }
else
     Z = B;
```

Такая двусмысленность особенно пагубна в ситуациях типа

```
if  (N > 0)
  for (I = 0; I < N; I++)
       if (S[I] > 0) {
   printf("...");
   return(I);
                    }
else   /* WRONG */
  printf ("ERROR - N IS ZERO\n");
```

Запись **else** под **if** ясно показывает, чего вы хотите, но компилятор не получит соответствующего указания и свяжет **else** с внутренним **if** (цикл **for** будет рассмотрен позже). Ошибки такого рода очень трудно обнаруживаются. Между прочим, обратите внимание, что в

```
if (A > B)
  Z = A;
else
  Z = B;
```

после Z=A стоит точка с запятой. Дело в том, что согласно грамматическим правилам за **if** должен следовать оператор, а выражение типа Z=A, являющееся оператором, всегда заканчивается точкой с запятой.

Упражнение 3:

Написать программу на языке С, которая вычисляет корни квадратного уравнения:

$$\mathbf{a}x^2 + \mathbf{b}x + \mathbf{c}=0,$$

где **a, b, c** – некоторые действительные числа.

Коэффициенты **a, b, c** вводятся с командной строки. Программа должна выдавать решение на экран.

Корни вычисляются по формуле:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

При этом:

если $b^2 - 4ac > 0$, то уравнение имеет два различных действительных корня;
если $b^2 - 4ac = 0$, то уравнение имеет один действительный корень кратности 2;
если $b^2 - 4ac < 0$, то уравнение действительных корней не имеет, а есть два комплексно-сопряженных корня:

$$x_1 = -\frac{b}{2a} + \frac{\sqrt{4ac - b^2}}{2a} i$$
$$x_2 = -\frac{b}{2a} - \frac{\sqrt{4ac - b^2}}{2a} i$$

Например, программа может выводить на экран строки:

$x_1 = -3 + 4i \quad x_2 = -3 - 4i$

или

$x_1 = -3 \quad x_2 = 4$

Обратить внимание, что при a=0 уравнение вырождается в линейное!

NB! На странице
http://www.gnu.org/software/libc/manual/html_node/Exponents-and-Logarithms.html
можно найти описания функции возведения в степень **pow** и вычисления квадратного корня **sqrt,** или можно воспользоваться командой **man <имя функции>**.

Лучше использовать переменные ***плавающего типа***, т.к. корни уравнения не всегда целые.

Структура программы может быть следующей:

```
.......
{
//декларирование переменных:
//коэффициенты  a,b,c
//вспомогательные переменные и результат
...
//вычисления и вывод на экран ...
if (temp > 0)
{
...
}
else
{
```

```
//  если temp < или 0  { ... то вывести на экран ... }
//  ...
...
}

//вывод на экран ...
...
}
```

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:

- ***Кернига́н 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: 27.09.2007 (1.10.2008)
Обновлено: 16.10.2008
Обновлено: 18.10.2010