

EL2310 – Scientific Programming

Lecture 5: Basics of C



Ramviyas Parasuraman (ramviyas@kth.se)

Royal Institute of Technology – KTH

Overview

Lecture 5: Basics of C

- Wrap Up

- Closer look at “Hello World”

- Programming Environment

- Basic Datatypes and printf

- Branching and Loops in C

- ▶ Last lecture on Matlab
 - ▷ Passing function as argument: `functionB(@functionA)`
 - ▷ Profiling: `profile on/off/viewer`
 - ▷ Debugging: Break points and step run
- ▶ Introduction to C
 - ▷ Roots of C
 - ▷ "Hello World" program

```
main()
{
    printf("Hello world\n");
}
```

- ▶ A C program consists of *functions* and *variables*
- ▶ Functions are built using statements
- ▶ Program execution starts in the function `main`
- ▶ Each program must have a `main` function

Analysis of the program

- ▶ Program starts with `#include <stdio.h>`
- ▶ Instructs the compiler to include information from the standard library for input and output (I/O)
- ▶ These lines are typically found at the top of the file
- ▶ The `main` function can, but does not have to have arguments
- ▶ The statements within a function should be placed between braces

The `printf` function

- ▶ `printf` is a command used to print to standard output
- ▶ The argument is a string (enclosed in double quotes)
- ▶ Will see later that it can take more arguments
- ▶ The last character in the string is `\n` which is C style for the newline character
- ▶ Other "hidden" characters can be obtained with an *escape sequence* (`\`)
- ▶ `\t` is a tab character

- ▶ Can be downloaded from the course materials page
- ▶ Ubuntu Linux guest running inside VirtualBox
- ▶ Preinstalled: gcc/g++/SDL/emacs
- ▶ VirtualBox can be installed in any host OS
- ▶ Go to: www.virtualbox.org, download and install
- ▶ Unpack the VM and use Machine-Add, then Start
- ▶ Use Shared Folders to exchange files with your host OS

Lecture 5: Basics of C

Wrap Up

Closer look at “Hello World”

Programming Environment

Basic Datatypes and printf

Branching and Loops in C

C statement

- ▶ A statement in C can be a single line followed by semicolon, or
- ▶ many statements enclosed by braces { }

Comments

- ▶ Single-line comments using //

- ▶ Multi-line comments

The compiler will ignore everything between /* and */

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

```
main()
```

```
{
```

```
    /* This is a not nice comment */
```

```
    printf("Hello world\n"); // This line prints
```

```
}
```

Data types

- ▶ There are only a few data types in C

`char`: character - a single byte

`int`: integer

`float`: floating point number

`double`: double precision floating point

- ▶ Can add qualifiers to get versions of these

`short int`: fewer bytes integer (maybe, depends on platform)

`long int`: integer with more bytes (maybe, depends on platform)

`unsigned int`: unsigned version (i.e. min value 0)

`signed int`: signed version (the default)

Variable declarations

- ▶ In Matlab no need to declare a variable
- ▶ In C you need to declare the variables before you use them
Can be as close to where they are used as possible
- ▶ **Syntax:** `<type> <variable_name>`
`int some_number;`
`int anumber, anothernumber, yetanothernumber;`
`int some_number=3;`

printf

- ▶ You can use `printf` to print not only for strings but the value of variables

Ex: `printf("This is iteration %d and the error is %f\n", iter, err);`

- ▶ To indicate that you want to print out a variable value you use the `%` character followed by a specification for what variable that is

`%d` to print integer

`%f` to print floating point

printf cont'd

- ▶ You can specify how many characters should be printed (at least)

```
printf("The number of participants is %6d\n",  
dist)
```

Will print at least 6 character

Ex: The number of participants is 4

- ▶ Can be used to align things

printf cont'd

- ▶ You can specify how many characters after a decimal point you want (at least)

```
printf("The distance is %.2fm\n", dist)
```

Will print 2 decimals

Ex: The distance is 4.00m

- ▶ Can combine number of characters and number of decimals

```
printf("The distance is %6.2fm\n", dist)
```

Will print 6 characters and 2 decimals

Ex: The distance is 4.00m

Notice that the dot counts as a character

- ▶ Can pad with zeros

```
printf("The distance is %06.2fm\n", dist)
```

Ex: The distance is 004.00m

printf cont'd

- ▶ More switches to printf
 - ▷ %O octal
 - ▷ %X hexadecimal
 - ▷ %C character
 - ▷ %S character string
 - ▷ %% to get % itself
- ▶ www.cplusplus.com/reference/clibrary/cstdio/printf/

Task 5.1

- ▶ Declare an integer and print this integer in decimal, octal and hexadecimal form

sizeof

- ▶ Different types have different sizes
- ▶ The function `sizeof` can be used to get the size, i.e. number of bytes of a variable or data type
- ▶ Syntax: `sizeof(<variable/data type>)`
- ▶ Is an operator not a function
- ▶ Relates data types to the Machine type

Task 5.2

- ▶ Write a program that lists the number of bytes for some of the basic data types
- ▶ Is there a difference between `short int`, `int` and `long int` on your machine?
- ▶ Do **NOT** assume the size of a type

Questions regarding the Matlab project?

Lecture 5: Basics of C

Wrap Up

Closer look at “Hello World”

Programming Environment

Basic Datatypes and printf

Branching and Loops in C

if-else

- ▶ Can control the flow with if-else

```
if (<expression>) <statement>
```

- ▶ or

```
if (<expression>)
```

```
    <statement>
```

```
else
```

```
    <statement>
```

- ▶ Remember that statement could be one line followed by semicolon
- ▶ or many lines with semicolon enclosed in { }
- ▶ Difference from MATLAB: The logical expressions have to be inside parentheses

if-else cont'd

- ▶ If you want to test more than one thing you can extend it with

```
if <expression>
    <statement>
else if <expression>
    <statement>
else
    <statement>
```

Logical expressions

- ▶ Similar to MATLAB
- ▶ Everything non-zero evaluates to true, zero is false
- ▶ Ex:

```
int value = 1;
if (value) {
    printf("It is true\n");
} else {
    printf("It is not true\n");
}
```

Simple manipulations

- ▶ Assign a value to a variable: `i = 0`
- ▶ Increment a variable: `i += 2;`
(which is short for `i = i + 2;`)
- ▶ If increment is 1 we can also write: `i++;`
`i--;` is the same as `i = i - 1;`
- ▶ Difference?: `i++` vs `++i`

switch

- ▶ Just like in matlab you can use `switch`
- ▶ Syntax:

```
switch (<variable>)  
{  
    case value1:  
        <statement>  
    break;  
    case value2:  
        <statement>  
    break;  
    default:  
        <statement>  
}
```

Task 5.3

- ▶ Write a program that generates a random number 0,1,2,...,9 and prints out a special message for 0 and 1 and a general message for 2-9.
- ▶ `stdlib.h`, `time.h`
`www.cplusplus.com/reference/clibrary/cstdlib/`
- ▶ Seed: `srand(seed)`, one can use current epoch time: `time(NULL)`
- ▶ Random number: `rand()` from 0 to `RAND_MAX` (at least 32767)
- ▶ Modulo (MATLAB `mod`): `%`

for-loop

- ▶ Can repeat code with `for`-loop

- ▶ Syntax:

```
for(<statement1>; <expression>; <statement2>)  
    <statement3>
```

- ▶ Typically:

```
for(variable=value1; <expression>; variable++)  
    <statement3>
```

- ▶ Need to declare `variable` and `value1` above

This can be done inside `for`

- ▶ `<expression>` is typically something that tests the value of the `variable` against some limits

- ▶ Ex:

```
for (i = 0; i < 10; i++)  
    printf("i=%d\n", i);
```

Task 5.4

- ▶ Write a program that loops over two variables until one reaches limit. The first one should go from 0 to 9 and the second from 42 to 60 with step 2
- ▶ Use operator , (coma)

while-loop

- ▶ **Syntax:** `while(<expression>) <statement>`
- ▶ `<expression>` is typically something that test the value of some variable changed inside the loop

- ▶ **Ex:**

```
while (i < 10) {  
    printf("i=%d\n", i);  
    i++;  
}
```

do-while-loop

- ▶ **Syntax:** `do <statement> while(<expression>)`
- ▶ `<expression>` is typically something that test the value of some variable changed inside the loop
- ▶ Will always execute the loop at least once!
- ▶ Ex:

```
i = 10;
do {
    printf("i=%d\n", i);
    i++;
} while (i < 10);
```

Task 5.5

- ▶ Write a program that prints a table with conversion from Celsius to Fahrenheit
- ▶ Tip: $F = 32 + (9/5) * C$

Homework

- ▶ Homework:
 - ▷ Install and run the virtual machine, or use:
 - Native Linux on your laptop
 - CSC computers
 - ▷ Use your editor
 - ▷ Try the tasks
 - ▷ Check out coding conventions!
- ▶ Wednesday: Continue with C