

EL2310 – Scientific Programming

Lecture 8: Basics of C



Yasemin Bekiroglu
(yaseminb@kth.se)

Royal Institute of Technology – KTH

Overview

Lecture 8: Basics of C

- Arrays

- Functions and return values

- Other tasks and useful stuff

- Strings

Assigning initial values to arrays

- ▶ You can assign values to the array when you declare them
- ▶ `int values[3] = {1, 2, 3};`
- ▶ You do not have to assign all values but you cannot assign too many
- ▶ You can also let the assignment define the number of elements
- ▶ `double matrix[] = {1, 2, 3, 4};`
will give you an array with 4 elements

Character arrays

- ▶ The most commonly used array in C is the character array

Ex: `char myname[32];`

- ▶ Assigning initial value to a character array:

`char myname[]="This is my name";`

Multidimensional arrays

- ▶ You can have more than one dimension in the array
- ▶ You add more `[]` at the end
- ▶ Ex: `double matrix[3][3];`
- ▶ You set/get elements using syntax `matrix[i][j]`

Assigning initial values to arrays cont'd

- ▶ For two dimensional arrays

```
double matrix[3][2] = {1, 2, 3, 4, 5, 6};
```

or a bit more clear

```
double matrix[3][2] = {{1, 2},  
                        {3, 4},  
                        {5, 6}};
```

- ▶ Can let assigned value define size (but only one of them!)
- ▶ `double matrix[][2] = {1, 2, 3, 4};`
will give you a 2x2 matrix

Task 2

- ▶ Write a program that multiplies two matrices and prints the result

Lecture 8: Basics of C

Arrays

Functions and return values

Other tasks and useful stuff

Strings

Functions

- ▶ Functions provide a way to encapsulate a piece of code
- ▶ Gives it a well defined input and output
- ▶ Makes code easier to read
- ▶ Often can assume the contents of a function based on its description

Functions, cont'd

▶ **Syntax:**

```
return-type function-name([arguments])  
{  
    declarations  
    statements  
}
```

- ▶ If the function does return anything you give it return-type `void`
- ▶ If you return something you leave the function with statement:
`return value;`
where `value` is of the return-type
- ▶ If the function has return-type `void` you leave with `return` if you want to leave before the function ends, otherwise you do not have to give an explicit `return`

Functions, cont'd

- ▶ NOTE: If your function has a return type and you do not have an explicit return the function will return something undefined.

return of main?

- ▶ `main` should return an `int`
- ▶ The return value can be read by whoever is calling `main` e.g. the OS
- ▶ When you have run a program in a *bash* shell you can see the return value in the special variable `$?`
- ▶ Ex:

```
./hello  
echo $?
```

Arguments to functions

- ▶ Can pass arguments into functions like in Matlab
- ▶ `double convert_to_fahrenheit(double tempC);`
- ▶ `double convert(double in, int type);`
- ▶ The arguments become independent local variables inside function

Declaring functions

- ▶ A function just like a variable need to be declared before it is used
 - ▷ Either put the definition of the function before it is used or,
 - ▷ add a declaration of it first and then later define it

- ▶ File example:

```
#includes
```

```
#defines
```

```
function declarations
```

```
main() { ... }
```

```
function definitions
```


Lecture 8: Basics of C

Arrays

Functions and return values

Other tasks and useful stuff

Strings

Task 3

- ▶ Write function that returns the probability to draw a certain value x given that it is from a normal distribution $\mathcal{N}(\mu, \sigma)$
- ▶ `double getprob(double x, double mean, double sigma);`
- ▶ Print a table with x and $p(x)$

Hint: You will have to include `<math.h>` and link with `libm (math)`

Linking to extra libraries

- ▶ Often use function defined in other libraries, such as `cos`, `sin`, `exp` from `libm`
- ▶ Need to tell linker that it should use `libm` as well
- ▶ `gcc -o mymathprg mymathprg.c -lm`

enum

- ▶ enumeration constant
- ▶ An alternative to using many `#define`
- ▶ Ex:

```
enum state { STATE_START, STATE_RUN, STATE_STOP};
```
- ▶ First name assigned value 0, next 1, etc
- ▶ The same with `#define`

```
#define STATE_START 0
#define STATE_RUN 1
#define STATE_STOP 2
```
- ▶ Can give value to all names manually
- ▶ Unassigned names will be assigned “last + 1”

Task 4

- ▶ Test enum
- ▶ What if you add as a last item `NUMBER_OF_ITEMS` in the enum?

Lecture 8: Basics of C

Arrays

Functions and return values

Other tasks and useful stuff

Strings

char array: C style strings

- ▶ Ex: `char name[] = "Tulou";`
- ▶ `strlen(...)` return length of a string
- ▶ A string is terminated by `\0`
- ▶ The variable `name` will be of length 6 where last character has value `\0`

Hint: You have to include `<string.h>`

Task 5

- ▶ Experiment with char arrays, strlen and sizeof
- ▶ What if `char [] name= "John Smith"`, what is the string length?
- ▶ What is the array size in bytes?
- ▶ What happens if you set `name[4] = 0;`

Relational operators

- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to
- == equal to
- != not equal to

Task 6

- ▶ What will the following do?

```
i = 4;
printf("i=%d\n", i);
if (i = 1) printf("i=%d\n", i);
printf("i=%d\n", i);
```

Assignment

- ▶ Assignment returns value
- ▶ Therefore, we can assign multiple variables
- ▶ Ex: `x = y = 0;`
- ▶ Assigns from right to left

Task 7

- ▶ Make the following expression clear by adding parentheses

`x = y = z = 4;`

Precedence

Incomplete table of precedence

- | | | | | | |
|----|----|----|-----|-----|---|
| 1. | () | [] | -> | . | |
| 2. | ! | ~ | -- | ++ | & |
| 3. | * | / | % | | |
| 4. | + | - | | | |
| 5. | > | >= | < | <= | |
| 6. | == | != | | | |
| 7. | && | | | | |
| 8. | | | | | |
| 9. | = | += | --= | ... | |

Evaluating logical expressions

- ▶ Logical expressions are evaluated left to right
- ▶ Guaranteed to stop as soon as expression value is determined
- ▶ A logical expression that evaluates to true is assigned value 1
- ▶ A logical expression that evaluates to false is assigned value 0

Task 8

- ▶ Write function `double atof(char s[])`
- ▶ Should take a char array as input and return a double representation of the string
- ▶ Assume that the string is a number like `-1.234` or `123.4`

Hint: Functions `isdigit`, `isspace` from `stdlib.h` are useful

<http://www.asciitable.com/>