

EEE243 – Applied Programming

Basic Programming Concepts

ROYAL MILITARY COLLEGE OF CANADA
ELECTRICAL & COMPUTER
ENGINEERING



GÉNIE ÉLECTRIQUE
ET GÉNIE INFORMATIQUE
COLLÈGE MILITAIRE ROYAL DU CANADA



Procedural programming

A programming paradigm based upon the concept of the procedure call [1].

Procedures are more commonly called functions and provide a way to organize a program and allows to reuse code without repeating it.

Procedural programming

Other programming paradigm:

- Object-oriented (seen in EEE320)
- Functional

Review of Basic Programming

- These are basic programming concepts you should have seen before
- Refer to the document “Just Enough C” provided for more details on the specific syntax of C
- Refer to the C coding style page on the website

Variables

Unlike in Matlab or Python, one must declare the type of the variable before using it.

Some basic C types:

- int: integers
- char: character
- float: floating point value

Variables

C	Matlab	Python
<pre>int x = 10 int y = -120</pre>	<pre>x = 10 y = -120</pre>	<pre>x = 10 y = -120</pre>
<pre>char a_char = 'a'</pre>	<pre>a_char = 'a'</pre>	<pre>a_char = 'a'</pre>
<pre>float a_float = 10 float another_float = 10.0</pre>	<pre>a_float = 10.0 another_float = 10.0</pre>	<pre>a_float = 10.0 another_float = 10.0</pre>

Scope and Extent

Scope: determines where a variable is visible.

Extent: determines the “duration” of a variable, that is, how long its value will exist for.

```
int a_function (int x, int z){  
    int y = x + z;  
    for (int i = 0; i < 10; i++) {  
        printf("i is %d", i);  
    }  
    return y;  
}
```

Operators

- Normal arithmetic operators have their usual precedence, so the expression $6+3*-7$ evaluates to -15 and is equivalent to $6+(3*(-7))$.
- Compound assignments: $+=$, $-=$, $*=$
- short form for incrementing and decrementing a variable: $++$ and $--$

Operators

- As in most other languages, the comparison operators are as follow:

==	is equal to (a == b is true if a is equal to b)
!=	is not equal to
<	is less than
>	is greater than
<=	is less than or equal to
>=	is greater than or equal to

Operators

- There are three boolean operators:

!	logical not or “negation.” ! a is true if a is not true; ! a is false if a is not false
&&	logical and. a && b is true if a is true and b is also true
	logical or. a b is true if a is true or if b is true, including the case where both are true

Control structures

Branching statements

C	Matlab	Python
<pre>if (x <= 10) { printf("x is smaller than 10"); }</pre>	<pre>if x <= 10 disp('x is smaller than 10'); end</pre>	<pre>if x <= 10 print("x is smaller than 10")</pre>
<pre>if (x <= 10) { printf("x is smaller than 10"); } else { printf("x is greater than 10"); }</pre>	<pre>if x <= 10 disp('x is smaller than 10'); else disp('x is greater than 10'); end</pre>	<pre>if x <= 10: print("x is smaller than 10") else: print("x is greater than 10")</pre>

Control structures

Branching statements

C	Matlab	Python
<pre>if (x <= 10) { printf("x is smaller than 10"); } else if (x >= 5) { printf("x is greater than 5"); } else { printf("x is not "between" "5 and 10"); }</pre>	<pre>if x <= 10 disp('x is smaller than 10'); elseif x >= 5 disp('x is greater than 5'); else disp('x is not between 5 and 10'); end</pre>	<pre>if x <= 10: print("x is smaller than 10") elif x >= 5: print(" is greater than 5") else: print("x is not between 5 and 10")</pre>

Control structures


Branching statements

C	Matlab	Python
<pre>switch (value) { case 5: printf("The value is five\n"); break; case 10: printf("The value is ten\n"); break; }</pre>	<pre>switch value case 5 disp('The value is five') case 10 disp('The value is ten') end</pre>	<pre>if value == 5: print("The value is five") elif value == 10 print("The value is ten")</pre>

Control structures


Loop statements

```
while (c > 100)
```



condition

```
for (int c = 200; c > 100; c--)
```



initialization condition update

Control structures

Loop statements

C	Matlab	Python
<pre>while(x < 10) { printf("x is %d\n", x+ +); }</pre>	<pre>while x < 10 disp(['x is ', num2str(x)]); x = x + 1; end</pre>	<pre>while x<10: print 'x is ', x x = x + 1</pre>
<pre>for(int i = 0; i < 10, i++) { printf("i is %d\n", i); }</pre>	<pre>for i = 1:10 disp(['i is', num2str(i)]); end</pre>	<pre>for i in range(0, 10): print 'i is ', i</pre>

Comments

C	Matlab	Python
<pre>/* Some block comments on multiple lines */</pre>	<pre>%{ Some block comments on multiple lines %}</pre>	<pre># Some block comments # on multiple lines</pre>
<pre>// Some inline comments</pre>	<pre>% Some inline comments</pre>	<pre># Some inline comments</pre>

Basic Structure of a C Program

```
/*
 * This is a demonstration program
 *
 * Author: Adrien Lapointe
 * Version: 28 Sept 2017
 *
 */
```

} File header comment

```
#include <stdio.h>
#include <stdlib.h>
#define A_CONST 10
```

} Pre-processor instructions

```
int a_function(int x);
```

} Function declaration

```
int gobal_var = 10;
```

} Global variable declaration

```
int main(void) {
    printf("Hello World!\n");
    printf("A defined constant: %d\n", A_CONST);
    printf("A global variable: %d\n", gobal_var);
    printf("a_function returns %d when given %d\n", a_function(3), 3);
    return EXIT_SUCCESS;
}
```

} main function

```
/*
 * Multiplies the parameter by two and returns the new value.
 *
 * x: a value
 * returns: the modified value
 */
```

} Function comment

```
int a_function(int x) {
    x *= 2;
    return x;
}
```

} Function definition

Functions

You could technically write all your code in the main function but you would lose on

- clarity
- readability
- reusability

Bottom line: **Use functions**

Functions

In C, functions need to be declared and defined

```
float a_function(int count, float weight);
```

return
type

function name

parameters

Functions

```
/*
 * This is a demonstration program
 *
 * Author: Adrien Lapointe
 * Version: 28 Sept 2017
 *
 */

#include <stdio.h>
#include <stdlib.h>
#define A_CONST 10

int a_function(int x); } Function declaration

int gobal_var = 10;

int main(void) {
    printf("Hello World!\n");
    printf("A defined constant: %d\n", A_CONST);
    printf("A global variable: %d\n", gobal_var);
    printf("a_funtion returns %d when given %d\n", a_function(3), 3);
    return EXIT_SUCCESS;
}

/*
 * Multiplies the parameter by two and returns the new value.
 *
 * x: a value
 * returns: the modified value
 */

int a_function(int x) { } Function definition
    x *= 2;
    return x;
}
```

function call

Functions

- Use functions to make your code more clear
- Give them names that are meaningful:
 - Ex.: `convert_mark` instead of `function1`

Functions

C	Matlab	Python
<pre>char convert_marks(int mark) { if (mark>=50){ return 'P'; } else { return 'F'; } }</pre>	<pre>function letter = convert_marks(mark) if mark>=50 letter = 'P'; else letter = 'F'; end end</pre>	<pre>def convertMarks(mark): if mark >= 50: return 'P' else: return 'F'</pre>

Coding Convention

This is a summary, refer to the website for more details

- Use proper indentation and line length
- All blocks must be surrounded by braces
- Include file comments AND function comments

Coding Convention

Names:

- Names must always be chosen to clearly indicate the purpose of the item named.
- **Variables and function parameter names:** lower-case, words separated with underscores, usually nouns `student_name`
- **Functions:** lower-case, words separated with underscores `average_grade`
`update_student_record`

Coding Convention

Names:

- **constants and macros:** upper-case, words separated with underscores `TICKS_PER_CM`

Resources

- On the lab page of the website
- Just Enough C
- C Reference Card

JUST ENOUGH C FOR 243

C is quirky, flawed, and an enormous success.
— Dennis Ritchie, co-designer of C

A C program is like a fast dance on a newly waxed dance floor by people carrying razors.
— Walsh Ravens, NetBSD developer

This guide summarizes the C programming language as used in EEE243 Applied Programming at the Royal Military College of Canada. It's more of a quick reference than a tutorial; it deliberately leaves things out, and it occasionally over-simplifies, but it's enough to get you started.

The guide occasionally refers to Joseph H. Silverman's *C Reference Card*, from www.rmc.ca/~brown/~jhs/. The Card provides more breadth but less explanation. For further information, see the course text or the Wikipedia articles on C and its syntax. In this guide C code is written like this. Anything written like this in a code sample would need to be replaced with real C code in a program.

Contents	Contents
C program structure	1
main program file	2
modules or libraries	2
header files	2
module implementation files	2
Preprocessor directives	2
#include	2
#define	2
#ifdef and #ifndef	2
Basic types	2
integer types (including bool and char)	2
unsigned integers	2
floating point types	2
void	2
Arrays	3
Pointers	3
NULL	3
Values and literal representations	3
numbers	3
truth values	3
characters	3
escape sequence character literals	3
array literals	3
character strings	3
Identifiers (names)	3
Variable and parameter declarations	3
const	3
static	4
enum definition and use	4
struct definition and use	4
typedef definition and use	4
Expressions	4
operators and precedence	5
assignment (=)	5
compound assignment (+=, *=, ...)	5
modules	5
integer division	5
increment and decrement (++, --)	5
comparison	5
Boolean operators (&&, , !)	5
array index expression	5
pointer operators (address and dereference)	5
pointer arithmetic	5
function call expression	6
ternary conditional expression (a ? b : c)	6
Statements	6
expression statements	6
block statement (sequence)	6
if and if-else statements (selection)	7
switch statement (selection)	7
while statement (selection)	7
do-while statement (loop)	7
for statement (loop)	7
break	7
continue	8
Functions	8
function prototype (declaration)	8
function definition	8
return statement	8
static variables	8
Comments	8
Memory management	9
sizeof	9
malloc	9
calloc	9
free	9
printf and sprintf	9
literal characters	9
escape sequences	9
value placeholders (conversion specifiers)	9
Copyright and use	10

1

8 September 2015

Questions?

References

[1] Procedural programming, Wikipedia,
accessed 21 August 2017,
https://en.wikipedia.org/w/index.php?title=Procedural_programming&oldid=776455885