ICT1002 – WEEK 8 – LAB

Introduction to C programming and development environments

1. OBJECTIVES

1. To choose, install, and configure a development environment suitable for C programming.
2. To comprehend C expressions and programs.
3. To write, compile, and debug C programs.

2. CHOOSING A DEVELOPMENT ENVIRONMENT

Unlike Python, C is not packaged with a standard development environment like IDLE. Instead, many different vendors have created C compilers for many different computing platforms, and an even greater number of vendors have created editors for C programs (these editors usually support many other languages as well). You may choose whichever compiler and editor that you like. This section describes a few well-known options that you can access for free.

2.1 WINDOWS

There are a few main free options for C development on Windows, e.g. Microsoft Visual Studio, MinGW, Code::Blocks, etc. There also numerous programmers' editors with support for C such as Notepad++, UltraEdit, and Atom, which can be used with a command-line compiler.

2.1.1 MICROSOFT VISUAL STUDIO

Visual Studio is popular amongst professional programmers and has many features designed to make working on larger programs easier, but can be resource-intensive and overwhelming at first. You can install the Community Edition for free on your own computer by visiting https://visualstudio.microsoft.com and following the instructions.

(i) The "Getting Started" page at https://visualstudio.microsoft.com/vs/getting-started/ provides several tutorials intended to introduce you to Visual Studio. Unfortunately, it is not obvious how to create a C program in Visual Studio even though it otherwise has good support for the language: you need to create an empty Visual C++ project (i.e. "Empty Project" under "Visual C++ > General" when creating a new project), then rename all of your source files to have a ".c" extension instead of the default ".cpp".

(ii) If you want to use Microsoft's compiler, but would rather use another editor, you can also access the command-line compiler ("cl") using the "Developer Command Prompt" available in the Start menu. The Developer Command Prompt is just normal command prompt with its environment set up to use the compiler tools, so that you can compile a C program called `hello.c` with

```
cl hello.c
```

This will create an executable file `hello.exe`, which can be run from the command line as usual. More complex programs made up of multiple .c files require some more complex operations, but this pattern should be enough to get us through today's lab.

2.1.2 MinGW

"MinGW" is a contraction of "Minimalist GNU for Windows". It provides a version of the GNU C Compiler along with the minimum number of other utilities required for it to work on Windows platforms. Once you've got MinGW installed, you should be able use it more or less the same way as you would use gcc on Unix or Linux.

You can download MinGW from http://www.mingw.org. Visit the HOWTO page at http://www.mingw.org/wiki/HOWTO for tutorials on installing and using MinGW. In order to complete this lab, you'll need to refer to:

- HOWTO Install the MinGW (GCC) Compiler Suite
- HOWTO Compile Programs with MinGW -- A Guide for New Users

2.2 MAC OS X

Apple provides a development environment for MacOS X called Xcode, which you can download from https://developer.apple.com/xcode/ (you'll need to have an Apple ID). Xcode includes the GNU C Compiler, a source code editor, and numerous other features that we do not need for this course.

See http://help.apple.com/xcode/mac/ for tutorials on developing applications using Xcode's integrated development environment. Ignore the examples that create graphical user interfaces – we only need console-mode programs for ICT1002. Alternatively, you can use gcc from the command line if you prefer a different editor, or just don't need all of the features of the integrated development environment. See the Linux section below for instructions on using gcc.

2.3 LINUX

Most Linux distributions have the GNU C Compiler (gcc) already installed. If not, use your package manager to install it. Most distributions also come with a text editor with some support for C programming, such as gedit (for GNOME-based desktops) or Kate (for KDE-based desktops). Many other editors exist if you want to try them.

To compile a program called `hello.c` from the command line, execute

```
gcc –o hello hello.c
```

This compiles and links `hello.c` to create an executable file called `hello`. You can then execute the program using

```
./hello
```

2.4 CROSS PLATFORM: CODE::BLOCKS

Code::Blocks is a *free cross-platform IDE for C, C++ and Fortran* built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable. You may download it from http://www.codeblocks.org/ and install it on your computer. You may also learn how to use it at: http://www.codeblocks.org/user-manual

2.5 ONLINE

You may also use on-line compilers like https://repl.it/ to test out your C programs.

3. EXERCISES FOR WEEK 8 LAB

WEEK_8_LAB_EXE_1: HELLO WORLD!

Once you've chosen and installed your development environment, make sure that you can type in, compile, and execute the famous "Hello World!" program:

```
#include <stdio.h>

int main() {

    printf("Hello world!\n");
    printf("Welcome to ICT1002!\n");

    return 0;
}
```

WEEK_8_LAB_EXE_2: C EXPRESSIONS

Suppose that the following variable declarations have been made in a program:

```
int a = -1, b = 2;
float x = 0.1;
float y = 1.5;
char c = 'p';
```

(a) Write a program to print the value for each of the following calculation to the screen.

a) a / b
b) a * b
c) (b * 3) % 4
d) x * a
e) x * y
f) y / x
g) c – 3

(b) Write a program to print the results.

h) printf("%4d", a);
i) printf("%04d", b);
j) printf("a/b = %d", a / b);
k) printf("%x", b);
l) printf("%.2f", y);
m) printf("%10.1f", x);
n) printf("c =\t%c", c);

WEEK_8_LAB_EXE_3: READING INPUT AND PERFORMING CALCULATIONS

The Body Mass Index (BMI) is calculated as follow:

$$BMI = \frac{weightInKilograms}{heightInMetres \times heightInMetres}$$

Create a new program called "bmi" that asks the user to type in his or her weight in kilograms and height in metres, then calculates and displays the user's BMI to one decimal place. The application should also evaluate whether the user is underweight or overweight according to the following table:

BMI	Evaluation
Less than 18.5	Underweight
Between 18.5 and 24.9	Normal
Between 25.0 and 29.9	Overweight
30.0 or greater	Obese

The following shows some sample output for the program, with the user input shown in red:

```
Enter your weight in kilograms: 65
Enter your height in metres: 1.85
Your BMI is 19.0. That is within the normal range.
```

4. WEEK_8_LAB_ASSIGNMENT: GUESSINT

Write a program called `guessint` that will play a simple two-player number-guessing game as follows:

1. The program will output "Player 1, enter a number between 1 and 1000:". If the player enters a number that is not in this range (inclusive), the program will output "That number is out of range." This repeats until Player 1 has entered a number that is in range.
2. Player 2 (who has not been watching Player 1) has ten rounds in which to correctly identify the number. At the start of each round, the program will output "Player 2, you have *n* guesses remaining." where *n* is the number of rounds left.
3. The program will output "Enter your guess:" and wait for Player 2 to enter a number.
    a. If Player 2's guess is too high, the program will output "Too high."
    b. If Player 2's guess is too low, the program will output "Too low."
    c. If Player 2's guess is correct, the program will output "Player 2 wins." and stop.
    d. If Player 2's guess is out of range, the program will output "That number is out of range." and return to Step 3. The number of guess should *not* be incremented.
4. If Player 2 has not guessed the number at the end of the tenth round, the program will output "Player 1 wins." and stop.

Some sample output is shown over the page, with the user input shown in red:

```
Player 1, enter a number between 1 and 1000:
1500
That number is out of range.
Player 1, enter a number between 1 and 1000:
500
Player 2, you have 10 guesses remaining.
Enter your guess:
750
Too high.
Player 2, you have 9 guesses remaining.
Enter your guess:
250
Too low.
Player 2, you have 8 guesses remaining.
Enter your guess:
500
Player 2 wins.
```

Note the following:

- Use macros (`#define`) to represent all constants, so that it is easy to change things like the number of guesses allowed, the highest number allowed, and so on.
- Include comment your code that explains what each section of it does.

OTHER RESOURCES

http://www.cplusplus.com/reference/clibrary/ has good documentation for the C Library. You can look up how to use functions like printf(), scanf(), etc. here. You may ignore the parts of the site referring to C++ (you will see C++ in ICT1009).

https://www.tutorialspoint.com/cprogramming/index.htm has a series of tutorials on aspects of C, covering nearly all of the topics in this course plus some more obscure topics like storage classes.

If you want more practice, you can try sites like Jutge.org (http://jutge.org) and HackerRank (https://www.hackerrank.com). These have many kinds of problems to try and on-line compilers that can check whether or not your program is correct. Be warned that many of the problems are quite hard for beginning programmers!