UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Computer Programming 143

Practical 1

2016

Aim of Practical 1:

a) To write simple C programs that do mathematical calculations.

b) Write a program containing simple decision making statements.

c) Read in numbers from the user.

"Divide by zero. Congratulations! Your math just destroyed a city."
http://math-fail.com/2010/06/never-divide-by-zero-ever.html

`+++ Divide By Cucumber Error. Please Reinstall Universe And Reboot. +++`
Hex (Hogfather), Terry Pratchett

Instructions

1. Attendance is **compulsory** for all the practical sessions of your assigned group. See the study guide for more details.
2. The last section (usually the last 30 minutes) of the practical will be used for a test.
3. If more than three tests have been missed for what ever reason, you will receive an **INCOMPLETE** for the subject. See the study guide for more details.
4. You must do all assignments **on your own**. Students are encouraged to help each other **understand** the problems and solutions, but each should write his/her own code. By simply copying someone else's code or solutions, you will not build an understanding of the work.
5. You are responsible for your own progress. Ensure that you understand the practical work. Check your work against the memorandum that will be posted on Wednesday afternoons on learn.sun.ac.za.
6. Use H:\CP143 as the Eclipse workspace folder for all projects but it is highly suggested that you use a **flash drive to backup** all work done.
7. Create a new project **for each assignment**. See *Creating an Eclipse Project* in Practical 0 for instructions on how to do just that.
8. Include a comment block at the top of each source file according to the format given. It must include the correct filename and date, your name and student number, the copying declaration, and the title of the source file.
9. **Indent your code correctly!** Making your code readable is not beautification, it is a time and life saving habit. Adhere to the standards (refer to the documents on SUNLearn). You can use Ctrl+A and then Ctrl+I to auto-indent.
10. Comment your code sufficiently well. It is required for you and others to understand what you have done.

Common Errors

Always **read** the error message thoroughly and try to correct it **before** asking for help. Look at the line number given in the error message, or double-click on the error message, as a clue to where to search. The **last** error given in this list is quite common, don't be caught!

- `error C2146: syntax error : missing ';' before identifier`
  - You have left out a semi-colon at the end of a statement.
- `error: expected declaration or statement at end of input`
  - Opening and Closing braces ({ and }) are not matched. Make sure you have a closing brace for each opening brace.
- `warning: format '%d' expects type 'int *', but argument 2 has type 'int'`
  - You did not include the & in `scanf`.
- `error: ???? undeclared (first use in this function)`
  - You have forgotten the quotes in a `printf` statement, or used a variable you did not declare. Make sure that you declare all the variables you use, and that you type variable names correctly. Remember, C is case sensitive.
- `warning: suggest parentheses around assignment used as truth value`
  - You have used '=' in stead of '==' in a boolean expression, e.g. in an **`if`** statement. For example:
    you typed `if (num1 = 0)` instead of `if(num1 == 0)`
- `error: 'else' without a previous 'if'`
  OR An **`if`** statement without an **`else`** does not work correctly
  - You have placed a semicolon after **`if`** statement, before the opening brace ('{') of the following code block. e.g. **`if`** `(x>5);`
- `error: expected '(' before '????'`
  - You did not include brackets around boolean expression, e.g. **`if`** `x==5` instead of **`if`** `(x==5)`.
- `warning: statement with no effect`
  `error: expected ';' before string constant`
  - You did not include brackets in `printf` statement, e.g. `printf "Hello";` instead of `printf("Hello");`.

Common Errors (cont.)

- You have changed your code, but the following message appears in the console when you compile:
  `Nothing to build for Assignment1A`
  - You did not save your code before compiling. Eclipse compiles the last code you saved, not necessarily what's on screen.
- `cannot open output file Assignment5B.exe: Permission denied`
  - You have tried to compile while your program was still running. Always wait for your program to finish running, or stop it using the red "Stop" button above the console, before compiling your code. If you get this error message, close and re-open Eclipse to continue working.

Question A

Goal:

Write, compile and run a program that reads in integer values and displays the sum. Modify the program to perform other mathematical functions

1. Create a project named **Assignment1A** and source file **Assignment1A.c**. Make sure that this is the only open project in the workspace before compiling the program.

2. Make sure that your source file has a comment block with the correct format containing your name, date, file name and description (refer to the code extract below).

3. Enter the following source code, and modify it with your details and the correct date:

```
/* Filename:       Assignment1A.c
 * Date:           2016/01/01
 * Name:           Doe J.J.
 * Student number: 12345678
 * ------------------------------------------------------
 * By submitting this file electronically, I declare that
 * it is my own original work, and that I have not copied
 * any part of it from another source, unless instructed
 * to do so in the assignment.
 * ------------------------------------------------------
 * Calculation Programming exercise
 * ------------------------------------------------------
 */
#include <stdlib.h>
#include <stdio.h>

int main()
{
  int num1;       // First number to be input by user
  int num2;       // Second number to be input by user
  int ans;           // Variable to store math result

  setbuf(stdout, 0);

  printf("Please enter the first integer:\n");
  scanf("%d",&num1);            // Read integer1

  printf("Please enter the second integer:\n");
  scanf("%d",&num2);            // Read integer2

  ans = num1 + num2;       // Summing
  printf("\n%d + %d = %d", num1, num2, ans);      // Print sum
```

```
  return 0;
}
```

4. Save, Compile and Run the program. Click in the console at the bottom of the screen to enter integers for the **`scanf`** statements.

5. Expand the program to include:

   - division,
   - modulus,
   - and the average.

   Let the second number that is entered be the divisor when doing division and modulus operations.

6. Your program output should have the same format as shown in the *Sample Output* in point 8. To print a % symbol use **`%%`** in the **`printf`** statement.

7. Note that division or modulus by zero causes a **run-time error**. Protect your program from this error by checking that the second number that is entered by the user is not zero. If it is zero, do not do those two operations, instead let your program give output as shown in *Sample Output 2*. To do this you will need **`if...else`** statements.

   Example of **`if...else`** syntax:

```
if (number1 > 5) {
   printf("Larger than 5\n"); //True condition
}
else {
   printf("Not larger than 5\n"); //False condition
}
```

8. Sample Ouput:
   Your program should have the same format as the examples. When the user enters the same input as that shown in these samples, your code should result in exactly the same output.

   Sample output 1: (straightforward input)

```
Please enter the first integer:
5
Please enter the second integer:
10
5 + 10 = 15
5 / 10 = 0
5 % 10 = 5
The average of 5 and 10 is 7
```

Sample output 2: ("catching" the zero input before it causes an error)

```
Please enter the first integer:
10
Please enter the second integer:
0
10 + 0 = 10
10 / 0 is not allowed
10 % 0 is not allowed
The average of 10 and 0 is 5
```

Sample output 3: (negative numbers)

```
Please enter the first integer:
-2
Please enter the second integer:
5
-2 + 5 = 3
-2 / 5 = 0
-2 % 5 = -2
The average of -2 and 5 is 1
```

9. Running the program with the debugger:

   Run the program as before but use the **Debugger** (little bug symbol) this time.

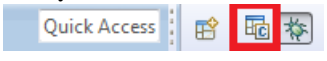
   The Eclipse environment will change from the C/C++ -perspective to the debug-perspective (left top window the execution thread information, right top the variables - default local, left bottom the source lines). The debug function is used to obtain detail of the inner workings of your program when the output is not what is expected. Note that the execution stop at the **`setbuf`** line (green).

```
17 int main()
18 {
19   int num1;       // First number to be input by user
20   int num2;       // Second number to be input by user
21   int ans;           // Variable to store math result
22
23   setbuf(stdout, 0);
24
25   printf("Please enter the first integer:\n");
26   scanf("%d",&num1);            // Read integer1
27
```

   Use the **F6** (step over) key to step one source line at a time. See how the variables change (after the line has been executed!!!). You still have to provide input.

   You can run more than one line by double-clicking on a future line and using the **F8** function key to resume from the current line. You can stop the debug by clicking the red square (**Ctrl+F2**).

   Switch the perspective back to the C/C++ after your debug run to edit code normally.



Note: *If your program "disappears" after the* **`setbuf`** *instruction (that is not in a debug library), rather set a breakpoint by double clicking on any standard future C source line after the* **`setbuf`** *instructions (and just* **left** *of the line number — a little circle with tick appears in the vertical blue bar) to set a breakpoint and then run (***F8***) to that point. You can still single step (***F6***) past normal C instructions.*

Something to think about: Your program discards the decimal part of the answers. Why?

Something to think about: In C and C++ the modulo operator % gives the REMAINDER after DIVISION and we can say that $B = A\%C$ is equivalent to $B = A - C \times \text{truncate}(A/C)$. Truncate is a mathematical operation which limits the number of digits after a decimal point. In that regard it is similar to rounding, but it differs from rounding in that it cuts the specified number of digits off without rounding up or down. Normally when the number of digits to be cut off is not specified, all digits after the decimal point are discarded.

Face the world: (This is extra information, if you do not understand it the first time, skip it and come back to it once you are finished with the practical.) In the world of programming the modulo operation gives the REMAINDER of DIVISION of one number by another, but this definition breaks down when either of the two numbers are negative. Programming languages differ in their definition/implementation of this operator. For example in C $(-2\text{mod}5) = -2$, while in Python $(-2\text{mod}5) = 3$. Matlab defines both of these definitions in the form of functions, REM and MOD. The reason for this disparity is in the type of division the language uses, which can be truncated division (see previous box), floored division or Euclidean division. Even though they give different results, all of the different implementations satisfy the mathematical modulo definition which says that:

$$B = A\text{mod}C \text{ when } (B - A) \text{ is an integer multiple of } C.$$

10. Ensure that you copy the **Assignment1A** project folder to a flash drive as a backup.

Question B

Goal:

Write a program that accepts an integer as input, determines if it is odd or even and then print a message to that effect.

1. Create a project named **`Assignment1B`** and source file **`Assignment1B.c`**. Make sure that this is the only open project in the workspace before compiling the program.
2. Make sure that your source file has a comment block with the correct format containing your name, date, file name and description (refer to the code extract in previous assignment).
3. Write a program that asks for and then reads an integer from the keyboard.
4. Test this number to determine if it is even or odd. Your program should work for positive and negative integers. An easy way to do this is to make use of the % operator (modulo). Consider the following examples:

$$\begin{aligned} 1\%2 &= 1 \\ 2\%2 &= 0 \\ -1\%2 &= -1 \\ -2\%2 &= 0 \end{aligned}$$

5. Print a message to the screen indicating whether the entered integer was even or odd. The format of your program output must look exactly like the *Sample output* of point 6.
6. Sample output :

Sample 1:
```
Enter an integer: 5
The integer is odd.
```

Sample 2:
```
Enter an integer: 3658
The integer is even.
```

Sample 3:
```
Enter an integer: -255
The integer is odd.
```

Sample 4:
```
Enter an integer: -898
The integer is even.
```

7. Ensure that you copy the **Assignment1B** project folder to a flash drive as a backup.