OBJECT ORIENTED PROGRAMMING

LAB 1 – REVIEW THE STRUCTURE OF A C/C++ PROGRAM. TESTING PROGRAMMING SKILLS. COMPARISON BETWEEN PROCEDURAL PROGRAMMING AND OBJECT ORIENTED PROGRAMMING

Course basics

The Object Oriented Programming course is focused on the C++ language.

We will be using the DevC++ environment available locally in the laboratory.

Structure of program

Both C and C++ share the following source code structure

- Include section – it includes both standard libraries (via < > separators) and custom libraries (via " " separators)
    - Using #include
    - Standard libraries are expected to be in common pathways found by the compiler
    - Custom libraries (usually part of your project) are in uncommon places
    - C++ standard libraries do not need the .h suffix in the file name
- Define section
    - Global macro definitions
    - Can use small letters, large letters and some symbols (including underscore character _)
- Namespace
    - The keyword "using namespace" are used to mark the beginning of a new namespace
- Global variables
- Function prototypes
- Function declarations
    - Including function bodies separated by { }
    - Exactly one of these (not more, not less) functions is called main, and it is the entry point in the program
    - Failing to have a function called main in your code can trigger errors in DevC++

Arrays vs vectors

- C uses arrays to store multiple data of a single type
    - Example of array

```
char * name = "John Smith";
```

- C++ introduces `vector, string,` as a special type of class-based variable.
- **As a good practice**, please do not use vector, string when referring to arrays both in writing and while speaking!

Classes vs structures

- Both share a similar internal composition
- struct can hold ONLY data members, although functions can be added indirectly by having data items of type pointer (pointing to a particular function) and its members are implicitly set with the public access modifier
- class can hold both data and function members, and its values are implicitly set with the public access modifier

Operators work the same way between C and C++.

- Assignment operator (=)
  - Example of use `x=5;`
- Arithmetic operators (+, -, *, /, %)

operator	description
+	addition
-	subtraction
*	multiplication
/	division
%	modulo

- Compound assignment (+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)
  - Can you think of examples of their use?
- Increment decrement (++, --)
  - You typically use them in for loops
- Relational and comparison operators (==, !=, >, <, >=, <=)

operator	description
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

- Logical operators (!, &&, ||)

operator	description
!	Not
&&	Logical/Boolean AND
||	Logical/Boolean OR

- Conditional ternary operator (? :)
  - Typically uses less assembler instructions than an if /else statement
  - *(condition) ? instruction_for_true : instructon_for_false;*
  - Example of use to compute max

```
max = (a>b) ? a : b;
```

- Comma operator (,)
  - Typical use case, in enumerations (like the arguments of a function)
- Bitwise operators (&, |, ^, ~, <<, >>)
  - Typical use cases when bit-bang-ing hardware registers in low level programming

operator	description
&	Bitwise AND
|	Bitwise inclusive OR
^	Bitwise exclusive OR
~	Unary complement (bit inversion)
<<	Shift bits left
>>	Shift bits right

- Cast operator
  - Typical use case, to change the type of a variable (required to solve some warnings in modern versions of C++)

```
int i;
float f;
i = (int) f; //changes the type from floating point to fixed point
```

- Sizeof operator
  - To return size of variable in memory (in bytes)
  - has variable **type** as parameter (not variable name)
  - Examples

`sizeof(int)` will return 4 (on most machines)

Basic input output.

- C++ uses iostream typically, and emplys cout to write to the console, cin to read from the console
  - Writing to console

```
cout << "Hello";
cout << 120;
cout << i;
```

- Reading from keyboard

```
int number;
cin >> number;
```

- cin.get() one way to keep the program running until you press a key

- C uses stdio.h usually
  - Writing to console
  - printf("%d, number);
  - Reading from keyboard – scanf always needs a pointer-type in argument 2!
  - scanf("%d", &number);
  - getch(); from conio.h to keep the program running until you keypress.

Testing your programming skills

It is assumed you have graduated the Computer Programming and the Data Structures and Algorithm classes.

As such, you should be able to achieve the following on your own, individually.

1. Write a simple Hello World example in either C or C++, your preference.

2. Solve the following problem.

Print the numbers from 1 to 100. For any number divisible with 3 print "fizz", for any number divisible with 5 print "buzz", instead of the number. In all the other cases print the number itself.

Have all values separated by a single space character, as separator.

Notes – to solve the exercise properly, the separator should not include a new line, tab or more or less than a single space. You should use the lower case letters for fizz and buzz.

Example

1 2 fizz 4 buzz [...] 14 fizzbuzz 16 […]

Comparison between procedural programming and object oriented programming

Procedural programming (C) relies on bodies of code called functions to implement functionality. Data and functions access and manipulation separate. Low means for access control. Really fast. Low memory footprint. Example of use: the Linux kernel.

Object oriented programming (C++, Java) rely on entities that combine data+functions, called classes (their definition), objects (each of their implementations). Better access control via access modifiers, both for data members and function members. Not so fast. Larger memory footprint. Example of use: Android framework.