Data Structures and Object Oriented Programming

Waqar S. Qureshi

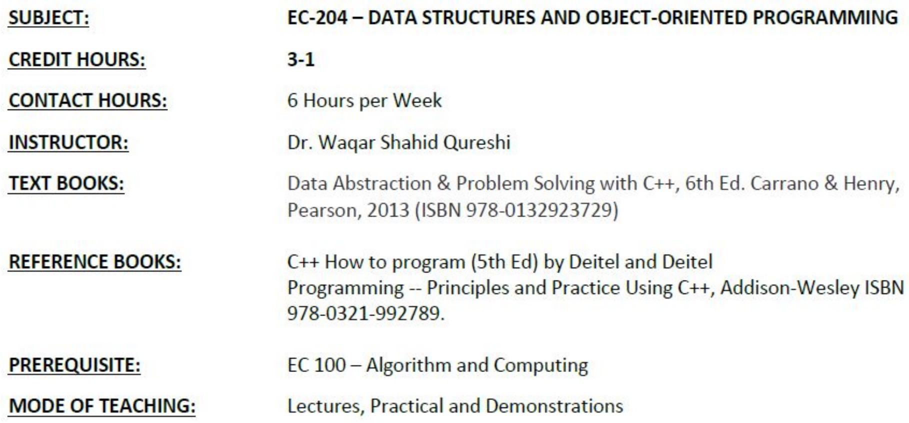
NUST College of Electrical & Mechanical Engineering.
waqar.shahid@alumni.ait.asia

September 24, 2017

Table of contents

This course

Topics to be covered

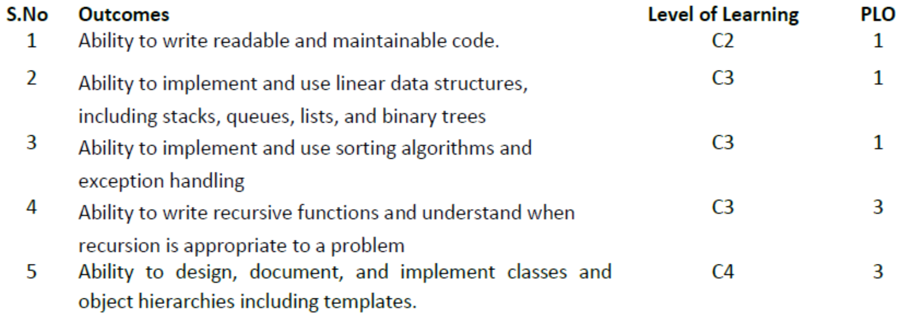


SUBJECT:	**EC-204 – DATA STRUCTURES AND OBJECT-ORIENTED PROGRAMMING**
CREDIT HOURS:	**3-1**
CONTACT HOURS:	6 Hours per Week
INSTRUCTOR:	Dr. Waqar Shahid Qureshi
TEXT BOOKS:	Data Abstraction & Problem Solving with C++, 6th Ed. Carrano & Henry, Pearson, 2013 (ISBN 978-0132923729)
REFERENCE BOOKS:	C++ How to program (5th Ed) by Deitel and Deitel Programming -- Principles and Practice Using C++, Addison-Wesley ISBN 978-0321-992789.
PREREQUISITE:	EC 100 – Algorithm and Computing
MODE OF TEACHING:	Lectures, Practical and Demonstrations

Topics to be covered

S.No	Topic	Week/Lecture
1	Overview of C++ and memory management	1-2
2	Objects and Classes	3
3	Self-Referential Structures	4-5
4	Linked Lists	5-6
5	Stacks and Queues	7
6	Trees	8-9
7	Sorting Algorithms	10
8	Operator Overloading	11
9	Inheritance	13
10	Polymorphism	14
11	Exception Handling	15
12	Templates	16

Class learning objectives



S.No	Outcomes	Level of Learning	PLO
1	Ability to write readable and maintainable code.	C2	1
2	Ability to implement and use linear data structures, including stacks, queues, lists, and binary trees	C3	1
3	Ability to implement and use sorting algorithms and exception handling	C3	1
4	Ability to write recursive functions and understand when recursion is appropriate to a problem	C3	3
5	Ability to design, document, and implement classes and object hierarchies including templates.	C4	3

C++ useful links

Those who like to revise the C++ programming please take a look at the reference book or online material available on the following website

- `https://www.tutorialspoint.com/cplusplus`
- `http://www.cplusplus.com/reference/`
- `http://www.cppreference.com/wiki/`

You can bring along your laptop and use the following online compiler to try code.

- `https://wandbox.org`
- `https://gcc.godbolt.org/#`
- `https://www.tutorialspoint.com/online_cpp_ide.php`

Overview of C++

Introduction

- C++ is a statically typed, compiled, general-purpose, case-sensitive, free-form programming language that supports procedural, object-oriented, and generic programming.
- C++ was developed by Bjarne Stroustrup starting in 1979 at Bell Labs in Murray Hill, New Jersey, as an enhancement to the C language and originally named C with Classes but later it was renamed C++ in 1983.
- C++ is a superset of C, and that virtually any legal C program is a legal C++ program.

The slides are prepared from material available at https://www.tutorialspoint.com/cplusplus

Object Oriented Programming

C++ fully supports object-oriented programming, including the four pillars of object-oriented development

- 1 Encapsulation
- 2 Data hiding
- 3 Inheritance
- 4 Polymorphism

Standard Libraries

Standard C++ consists of three important parts

- The core language giving all the building blocks including variables, data types and literals, etc.
- The C++ Standard Library giving a rich set of functions manipulating files, strings, etc.
- The Standard Template Library (STL) giving a rich set of methods manipulating data structures, etc.

Environment Setup

Environment Setup

To write C++ code you need an editor, example, nodepad++, gedit, VIM, EMAC, etc.

You can also use and IDE such as Visual Studio or Eclipse to write and manage your source code

To compile we need a compiler such as Visual Studio, GNU GCC or etc.

An executable code can be run using command line or using IDE

You can use both Windows or Linux Operating System.

I prefer *Linux*!

Basic Syntax C++

Basic Syntax C++

A C++ program can be defined as a collection of objects that communicate via invoking each other's methods.

Object

Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors - wagging, barking, eating. An object is an instance of a class.

class

A class can be defined as a template/blueprint that describes the behaviors/states that object of its type support.

Basic Syntax C++

Method

A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

Instance Variables

Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

Program Structure

Example (Hello DE-38)

```
#include <iostream>
using namespace std;
// main() is where program execution begins.
int main() {
   cout << "Hello DE-38!"; // prints Hello DE-38!
   return 0;
}
```

Program Structure

The C++ language defines several headers, which contain information that is either necessary or useful to your program. For this program, the header `<iostream>` is needed

The line using `namespace std;` tells the compiler to use the `std` `namespace`. Namespaces are a relatively recent addition to C++.

The next line "// main() is where program execution begins" is a single-line comment available in C++. Single-line comments begin with // and stop at the end of the line.

Program Structure

The line int `main()` is the main function where program execution begins.

The next line `cout << "Hello DE-38";` causes the message "Hello DE-38" to be displayed on the screen.

The next line `return 0;` terminates `main()` function and causes it to return the value 0 to the calling process.

Compile and Execute C++ Program

Open a text editor and add the code as above.

- Save the file as: `hello.cpp`
- Open a command prompt and go to the directory where you saved the file
- Type `g++ hello.cpp` and press enter to compile your code. If there are no errors in your code the command prompt will take you to the next line and would generate a.out executable file.

Compile and Execute C++ Program

Example

```
$ g++ hello.cpp
$ ./a.out
Hello DE-38!
```

Semicolons and Blocks in C++

A semicolon `;` is a statement terminator!

A block is a set of logically connected statements that are surrounded by opening and closing braces {}

C++ does not recognize end of line as statement terminator.

C++ is a case sensitive language

C++ does not allow punctuation characters such as @, $, and % within identifiers

C++ keywords may not be used as constant or variable or any other identifier names

Whitespace is the term used in C++ to describe blanks, tabs, newline characters and comments

C++ data types

C++ premitive datatype

Following table list down the seven primitive datatypes:

Type	keyword
Boolean 1	`bool`
Character	`char`
Integer	`int`
Floating point	`float`
Double floating point	`double`
Value less	`void`
wide character	`wchar_t`

Table: C++ primitive datatype

C++ datatypes

The basic types can be modified using the following modifiers

- signed
- unsigned
- short
- long

Sample code to know size of datatypes

Example

```
#include <iostream>
using namespace std;

int main() {
  cout <<"Size of char: " << sizeof(char) << endl;
  cout <<"Size of int: " << sizeof(int) << endl;
  cout <<"Size of short int: "<< sizeof(short int) << endl;
  cout <<"Size of long int: "<< sizeof(long int) << endl;
  cout <<"Size of float: "<< sizeof(float) << endl;
  cout <<"Size of double: "<< sizeof(double) << endl;
  cout <<"Size of wchar_t: "<< sizeof(wchar_t) << endl;
   return 0;
}
```

typedef Declarations and Enumerated Types

You can create a new name for an existing type using `typedef`.

An enumerated type declares an optional type name and a set of zero or more identifiers that can be used as values of the type. Each enumerator is a constant whose type is the enumeration. Creating an enumeration requires the use of the keyword enum

typedef declaration

```
[typedef type newname;]

typedef int feet;
feet distance;
```

Enumeration

```
enum color { red, green, blue } c;
c = blue;
```

Storage Classes in C++

Storage Classes in C++

A storage class defines the visibility and life-time of variables and/or functions within a C++ Program.

These specifiers precede the type that they modify.

There are following storage classes, which can be used in a C++ Program

- `auto`
- `register`
- `static`
- `extern`

Example Code - Use of static keyword

In C++, when `static` is used on a class data member, it causes only one copy of that member to be shared by all objects of its class:

```
#include <iostream>
 // Function declaration
void func(void);

static int count = 10; /* Global variable */

main() {
   while(count--) {
      func();
   }

   return 0;
}

// Function definition
void func( void ) {
   static int i = 5; // local static variable
   i++;
   std::cout << "i is " << i ;
   std::cout << " and count is " << count << std::endl;
}
```

Use of extern keyword - example

The `extern` modifier is most commonly used when there are two or more files sharing the same global variables or functions

First File: main.cpp

```
#include <iostream>

int count ;
extern void write_extern();

main() {
   count = 5;
   write_extern();
}
```

Use of extern keyword - example (continue...)

Here, `extern` keyword is being used to declare count in another file

Second File: Support.cpp

```
#include <iostream>

extern int count;

void write_extern(void) {
   std::cout << "Count is " << count << std::endl;
}
```

Now, compile the program and RUN on command line as follows:

```
$g++ main.cpp support.cpp -o write
$./write
5
```

Operators in C++

Operators in C++

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. Operators are categorized as follows:

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Misc Operators

Operators Precedence in C++

Category	Operator	Associativity
Postfix	$()[] - > . + + - -$	Left to right
Unary	$+ - !\ + + - - (\mathit{type}) * \mathit{sizeof}$	Right to left
Multiplicative	$* / \%$	Left to right
Additive	$+ -$	Left to right
Shift	$<< >>$	Left to right
Relational	$< <= > >=$	Left to right
Equality	$== != $	Left to right
Bitwise AND	$\&$	Left to right
Bitwise XOR		Left to right
Bitwise OR	$\vert$	Left to right
Logical AND	$\&\&$	Left to right
Logical OR	$\vert\vert$	Left to right
Conditional	$? :$	Right to left
Assignment	$= + = - = * = / = \% = >> = << = \& =^{=} \vert =$	Right to left
Comma	$,$	Left to right

Table: Operators Precedence in C++

Loop types in C++

Loop types in C++

A loop statement allows us to execute a statement or group of statements multiple times.

Following is the general from of a loop statement in most of the programming languages

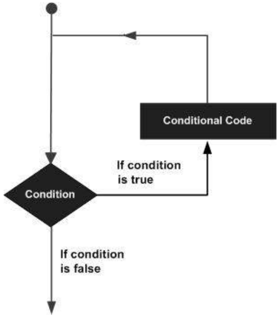


Figure: Ref: Tutorialspoint.com

Decision Making in C++

Loops types in C++

C++ have following loop types

- `while` loop
- `for` loop
- `do while` loop
- nested loop

C++ supports the following control statements.

- `break` statement
- `continue` statement
- `goto` statement (never use this!)

Loop types in C++ - Example

A loop becomes infinite loop if a condition never becomes false. Software programmer usually use for loop or while loop.

Infinite loop - Example

```
#include <iostream>
using namespace std;
int main () {
   for( ; ; ) {
      printf("This loop will run forever.\n");
   }
   return 0;
}
```

Some control system is usually used in embedded program to terminate!

Decision making statement in C++

Frame Title

Decision making structures require that the programmer specify one or more conditions to be evaluated or tested by the program.

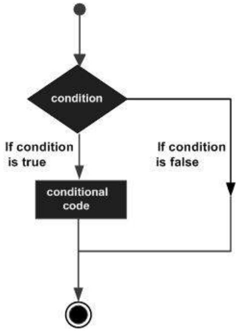


Figure: Ref: Tutorialspoint.com

C++ programming language provides, `if` statement, `if else` statement, and `switch` statement.

Function in C++

Function in C++

A function is a group of statements groups together to perform a task

A function's declaration, which tells the parameters and return type

A function's definition tells what the function will do

C++ standard library provides many of built-in function

The general form of a function

```
return_type function_name( parameter list ) {
   body of the function
}
```

function parameters type

Call type

1. call by value
2. call by pointer
3. call by reference

Call Description

1. This method copies the actual value of an argument into the formal parameter of the function.
2. This method copies the address of an argument into the formal parameter
3. This method copies the reference of an argumen into the formal parameter

Array in C++

Arrays in C++

We can make an array of any datatype. A specific element in array is accessed by an index.

Example of an Array

```
#include <iostream>
using namespace std;

#include <iomanip>
using std::setw;

int main () {

   int n[ 10 ]; // n is an array of 10 integers

   // initialize elements of array n to 0
   for ( int i = 0; i < 10; i++ ) {
      n[ i ] = i + 100; // set element at location i to i + 100
   }
   cout << "Element" << setw( 13 ) << "Value" << endl;

   // output each array element's value
   for ( int j = 0; j < 10; j++ ) {
      cout << setw( 7 )<< j << setw( 13 ) << n[ j ] << endl;
   }

   return 0;
}
```

Passing Array as argument

Example - Average function

```
double getAverage(int arr[], int
size)  int i, sum = 0; double
avg;
for (i = 0; i < size; ++i)  sum
+= arr[i];
avg = double(sum) / size;
return avg;
```

Example - main() function

```
include <iostream> using namespace
std;
// function declaration: double
getAverage(int arr[], int size);
int main ()
// an int array with 5 elements.
int balance[5] = 1000, 2, 3, 17,
50;
double avg;
// pass pointer to the array as an
argument.
avg = getAverage( balance, 5 ) ;
// output the returned value
cout << "Average value is: " << avg
<< endl;
return 0;
```

Strings in C++

Example of strings in C++

C++ supports two types of string representation:

- The C-style character array string.
- The string class type introduced with Standard C++.

```
#include <iostream>
#include <cstring>
using namespace std;
int main () {
   char str1[10] = "Hello";
   char str2[10] = "World";
   char str3[10];
   int  len ;
```

```
   // concatenates str1 and str2
   strcat( str1, str2);
   cout << "strcat( str1, str2): "
   << str1 << endl;

   // total lenghth of str1
   len = strlen(str1);
   cout << "strlen(str1): " << len << endl;

   return 0;
}
```

Example: using string class

string Example

```
#include <iostream>
#include <string>
using namespace std;
int main () {

   string str1 = "Hello";
   string str2 = "World";
   string str3;
   int  len ;

   // concatenates str1 and str2
   str3 = str1 + str2;
   cout << "str1 + str2 : " << str3 << endl;

   // total length of str3 after concatenation
   len = str3.size();
   cout << "str3.size() :  " << len << endl;
   return 0;
}
```

Pointers in C++

Pointers in C++

A pointer is an address of a memory location or an address of a variable in memory. Look at the following example:

Example: address of variable

```
#include <iostream>

using namespace std;
int main () {
   int  var1;
   char var2[10];

   cout << "Address of var1 variable: ";
   cout << &var1 << endl;

   cout << "Address of var2 variable: ";
   cout << &var2 << endl;

   return 0;
}
```

```
Address of var1 variable: 0xbfebd5c0
Address of var2 variable: 0xbfebd5b6
```

What are pointers

A pointer is a variable whose value is the address of another variable. You must declare a pointer before you can work with it.

```
type *var-name;
```

Different concepts in pointers include the following:

- NULL pointer and Pointer arithmetic
- Pointer vs Arrays
- Pointer to Pointer
- Array to pointer
- Passing pointer to function
- Return pointer from function

See footnote for revision[1]

[1] https://www.tutorialspoint.com/cplusplus/cpp_pointers.htm

References in C++

References vs pointers

What is the difference between references and pointers?

- You cannot have NULL references. You must always be able to assume that a reference is connected to a legitimate piece of storage.
- Once a reference is initialized to an object, it cannot be changed to refer to another object. Pointers can be pointed to another object at any time.
- A reference must be initialized when it is created. Pointers can be initialized at any time.

Read more about *references as parameters* and *reference as return value* on the website[2]

[2] urlhttps://www.tutorialspoint.com/cplusplus/cpp_references.htm

Date & Time in C++

Date and Time in C++

C++ inherits the structures and functions for date and time manipulation from C

To access date and time related functions and structures, you would need to include ¡ctime¿ header file in your C++ program.

See the footnote[3] to read more about the available structures and function to use time.

[3] https://www.tutorialspoint.com/cplusplus/cpp_date_time.htm

Date & time Example

Example: Current data and time

```
#include <iostream>
#include <ctime>

using namespace std;

int main() {
   // current date/time based on current system
   time_t now = time(0);

   // convert now to string form
   char* dt = ctime(&now);

   cout << "The local date and time is: " << dt << endl;

   // convert now to tm struct for UTC
   tm *gmtm = gmtime(&now);
   dt = asctime(gmtm);
   cout << "The UTC date and time is:"<< dt << endl;
}
```

Data Structures in C++

Multiple Columns

structure is an user defined *data type* which allows you to combine data items of different kinds

Suppose you want to save an *Image* in a memory. You need the following information about an image to be saved along with the data.

- width of image
- height of image
- Pointer to the image data

```
struct Image {
/* # of rows in image*/
int Rows;
/*# of columns in image */
int Cols;
/* Pointer to the image data*/
unsigned char *Data;
};
```

You can use the `typedef` keyword to alias `struct`

References

C++ Overview (2017)
Website
https://www.tutorialspoint.com

The End