# The Free C++ Advanced Concepts Guide

Robin Broad
B.Sc., M.Sc.

Free software and documentation
published under the GNU GPL

www.robinbroad.co.uk

**The Free C++ Advanced Concepts Guide**

Cover photo by Danny Meneses.
(https://www.pexels.com/@danny-meneses-340146)
Photo generously released under the Creative Commons Zero (CC0) license.
(https://creativecommons.org/publicdomain/zero/1.0/)

C++ logo created by Jeremy Kratz, licensed by The Standard C++ Foundation.
(https://isocpp.org/home/terms-of-use)

GPL 3 logo used with permission from the Free Software Foundation.
(https://www.gnu.org/graphics/license-logos.en.html)

This guide and associated programs are freely available online in all of the major English speaking countries of the world.

## About This Book

This book was written to help students to understand some of the advanced concepts of object orientated programming in C++. This book does not start with the details of getting started in C++, or the basics of controlling program flow. Rather, it starts off with the advanced concepts of classes and objects, inheritance, polymorphism, arrays and pointers.

The advanced concepts are all demonstrated using working C++ programs. The programs have been heavily commented to give students a step by step explanation for every line of code and every new concept. The code has been formatted using an easy to follow colour scheme specially designed for C++ code listings.

Importantly, all of the programs have been released as free software under the GNU GPL and can be downloaded from:
http://www.robinbroad.co.uk/

The free C++ program files are:

- ClassesAndObjects.cpp
- InheritanceAndPolymorphism.cpp
- ArraysAndPointers.cpp

This free advanced concepts guide was written by Robin Broad, a computer scientist and teacher from Newcastle upon Tyne in the UK. Robin achieved a distinction in his masters degree in computing science (MSc), which he studied at Newcastle University, in England from 2007 to 2008.

The decision to publish this book as a free advanced concepts guide under the GNU General Public License was inspired by the example of the American computer scientist Richard Stallman**.** He started the **GNU Project** in 1983, which had the goal of creating a "complete Unix-compatible software system" composed entirely of free software. Work began in 1984. Later, in 1985, Stallman started the Free Software

Foundation and wrote the GNU General Public License (GNU GPL) in 1989.

The GNU Free Documentation License (GFDL) is used for tutorials, reference manuals and other large works of documentation. It's a strong copyleft license for educational works, initially written for software manuals, and includes terms which specifically address common issues that arise when those works are distributed or modified.

This book was written using LibreOffice, an open source word processor, running on a GNU/ Linux computer, an open source operating system. GNU/Linux and its applications are a prominent example of **free** (to share, study and modify) **software**. This keeps us **free from licenses, patents** and **agreements, reduces costs** and **improves** the **reliability** of our systems. Free software has become the foundation of a learning society where we share our knowledge in a way that others can build upon and enjoy.

*Richard Stallman - American computer scientist, founder and president of the Free Software Foundation and author of the GNU General Public License (GNU GPL). Photo courtesy of Free Software Foundation, Inc.*

*Robin Broad B.Sc., M.Sc. - British computer scientist and teacher. Founder and Technical and Creative Director of Starbird Digital web services, Automated Teaching Machines and the author of this book.*

**Acknowledgements**

I would like to thank the following people for their support during the writing of this book and development of the associated software:

My friends and family for their encouragement and support during the development of the content and software. Newcastle University for allowing me to continue to work in their library, where much of this project was developed.

Richard Stallman who started the GNU Project in 1983, which led to the GNU/Linux environment that was used to build the website and write this book. he also created the GNU General Public License (GNU GPL) and the GNU Free Documentation License.

This service would not exist if it were not for the pioneers of networking, the ARPANET, the Internet and the World Wide Web, namely Claude Shannon, Vannevar Bush, Paul Baran, Donald Davies, Joseph Licklider, Charles Herzfeld, Bob Taylor, Douglas Engelbart, Larry Roberts, Leonard Kleinrock, Louis Pouzin, John Klensin, Bob Kahn, Vint Cerf, Steve Crocker, Jon Postel, Jake Feinler, Peter Kirstein, Danny Cohen, Paul Mockapetris, Joyce Reynolds, David Clark, Dave Mills, Radia Perlman, Dennis Jennings, Steve Wolff, Van Jacobson, Ted Nelson, Tim Berners-Lee, Mark McCahill, Robert Cailliau, Marc Andreessen and Eric Bina.

This work is dedicated to Chris and Barbara.

# Contents

# Section One
## A Brief History of Object Orientated Programming

## A Brief History of Object Orientated Programming

**Fortran** (Backus, IBM, New York, **1957**), short for mathematical FORmula TRANslation, was the first widely used *high-level computer language* to use a *compiler* to produce *machine code*, making programming easier. Fortran was most popular during the 1950s and 1960s. It included the modern concept of *data abstraction* (*information hiding*). The *floating point data type* had it's internal mechanism hidden from the programmer.

In 1967, **Simula 67** (Nygaard & Dahl, Norwegian Computing Center, Oslo, **1967**) was the world's first *object orientated programming* (OOP) language. It used *classes*; data structures with data and functions packaged together.

**Smalltalk** (Kay, Ingalls & Goldberg, Xerox PARC, California, **1972**) was a programming language designed so that objects could be communicated with by passing messages. The Smalltalk team were the first to introduce the term *object orientated programming* (OOP). Smalltalk was strongly influenced by **Lisp** (McCarthy, MIT AI Labs, Cambridge, Massachusetts, **1958**), short for LISt Processor, which used lists of objects/ atoms.

The **C programming language** (Ritchie, AT&T Bell Labs, New Jersey, **1972**) was written to re-write a portable version of the Unix operating system (originally written in assembly language) in C, a *high-level computer language* which uses a *compiler* to produce *machine code*. **C++** (Stroustrup,  AT&T Bell Labs, New Jersey, **1979**) extended the C programming language by adding Simula 67-like features to it, creating a powerful *object orientated programming language;* it was originally named C with classes. C++ was first standardised by the ISO (International Organization for Standardization ) in 1998.

*Bjarne Stroustrup, the creator of C++*
*(Image Permission GNU Free Documentation License)*

# **Section Two**

# C++ Classes and Objects

## C++ Classes and Objects

This program demonstrates classes, objects, constructors, destructors and member functions in C++

- Classes - A class is a data type defined by the programmer, in this case Cuboid. The class has members (dimensions a,b and c) and member functions to return useful information about the object. The class describes the structure of all objects of that class.

- Objects - An object is an instance of a class. When an object is created (constructed or instantiated), memory is allocated to it.

- Constructor - A constructor is a member function used to initialise an object when it is created. The constructor is public and returns no value.

- Destructor - A member function used to remove the object from memory when it is no longer needed.

- Member functions - class functions that return useful information about objects of the class, in this case volume, surface area etc.

The following code listing is heavily commented to give students a step by step explanation for every line of code and every new concept. The following code is a complete working C++ program which should compile and run without error.

This program is free software published under GNU GPL and can be downloaded free from http://www.robinbroad.co.uk/

```
/* C++ Classes and Objects Demo
Copyright © Robin Broad 2018
Free software published under GNU GPL
http://www.robinbroad.co.uk/
April 2018
*/

//Preprocessor instructions to the C++ compiler
//Include the standard C input/ output library
#include <iostream>
//Use the standard names in namespace std for the functions in the iostream library (e.g. cout).
Without this, the compiler would not recognise the function cout, which we are using in this program
using namespace std;
//Include the standard C maths library. We need it to calculate the square root sqrt() in the
lenSpaceDiag() member function of the Cuboid class
#include<math.h>

class Cuboid
/*Class name Cuboid. This class represents the three dimensional mathematical shape rectangular
cuboid. It has six rectangular faces, 12 edges and 8 vertices. We will provide member functions for
surface area, volume and the length of the space diagonal.
This code is heavily commented to assist students of C++
Author: Robin Broad
C++ Examples
http://www.robinbroad.co.uk/
April 2018
*/
{
        //*** Members ***
        private:
        /*private means accessible only within the class (or by friends of the class); not publicly
accessible.
        (This is also known as data hiding. This is part of the process of encapsulation;
information hiding, data and methods)*/

        //Declaring the class members a, b and c.
        int a, b, c;
        /*Using the standard mathematical notation of a, b and c for the lengths of the sides of the
cuboid
        int means "of type integer"*/

        //***Constructor ***
        /*The constructor is a member function with the same name as the class.
        It is called whenever an object (class instance) is created*/
        public:
        //public means these members or member functions can be accessed anywhere, including from
outside the class

        //The Cuboid class constructor member function. This is passed the cuboid dimensions a, b
and c.
        Cuboid(int fpa, int fpb, int fpc)
        //fpa=function parameter a etc.
        {
                /*Notifying us that the constructor has been called
                The insertion operator << specifies the string to be written to the console by cout
                The \n is the newline character*/
                cout << "Constructing a cuboid object of dimensions "<<fpa<<" x "<<fpb<<" x
"<<fpc<<" units\n";
                //Allocate the passed values to the private member variables of the object (class
instance)
                a = fpa;
                b = fpb;
                c = fpc;
        }
        //*** Destructor ***
        //The Cuboid class destructor member function. Destructs the object when it is no longer
needed.
        //The tilde (~) symbol means "not".
        ~Cuboid()
        {
                //notifying us that the object is being destructed
                cout << "Destructing a cuboid object of dimensions "<<a<<" x "<<b<<" x "<<c<<"
units\n";
        }

        //*** Member Functions ***
        //Defining the member functions for the Cuboid class

        int surfaceArea()
        /*Member function to calculate the surface area of the cuboid object
        Returns the answer as an integer. This in an inline member function (i.e. declared within
the class)
        This could have been declared outside the class (outline member function) using the form:
        int Cuboid::surfaceArea() {...}
        (where :: represents the scope resolution operator)
        */
        {
                return 2*(a*b+a*c+b*c);
                /*surfaceArea = 2 x (a x b + a x c + b x c)
                This is a standard mathematical concept*/
        }

        int volume()
        /*Member function to calculate the volume of the cuboid object
        Returns the answer as an integer. This in an inline member function (i.e. declared within
the class)
        This could have been declared outside the class (outline member function) using the form:
        int Cuboid::volume() {...}
        (where :: represents the scope resolution operator)
        */
        {
                return a*b*c;
                /*volume=a x b x c
                This is a standard mathematical concept*/
        }

        float lenSpaceDiag()
        /*Member function to calculate the length of the space diagonal of the cuboid object
        Note: returns the answer as a float as square root may not be an integer
        This in an inline member function (i.e. declared within the class)
        This could have been declared outside the class (outline member function) using the form:
        int Cuboid::lenSpaceDiag {...}
        (where :: represents the scope resolution operator)
        */
        {
                return sqrt(a*a+b*b+c*c);
                /*Length of space diagonal = square root(a^2 + b^2 + c^2)
                This is a standard mathematical concept*/
        }
};
//}; marks the end of the class definition
```

```
//*** Main program ***
int main()
{
        //Output Header
        cout <<"C++ Classes and Objects Demo\nRobin Broad\nMay 2018\n";
        /*This program demonstrates classes, objects, constructors, destructors and member functions
in C++
        Classes - A class is a data type defined by the programmer, in this case Cuboid. The class
has members (dimensions a,b and c) and member functions to return useful information about the
object. The class describes the structure of all objects of that class.
        Objects - An object is an instance of a class. When an object is created (constructed or
instantiated), memory is allocated to it.
        Constructor - A constructor is a member function used to initialise an object when it is
created. The constructor is public and returns no value.
        Destructor - A member function used to remove the object from memory when it is no longer
needed.
        Member functions - class functions that return useful information about objects of the
class, in this case volume, surface area etc.*/

        //Create a cuboid object of dimensions 4 x 5 x 2
        //Objects can contain organised groups of data (in this case the dimensions of the cuboid)
and member functions which return useful results about the object
        Cuboid cuboidA(4, 5, 2);
        //Create a second cuboid object
        Cuboid cuboidB(8, 12, 4);
        //Create a third cuboid object
        Cuboid cuboidC(5, 10, 3);
        //Call the member function surfaceArea() of cuboidA to find it's Surface area
        cout <<"Surface area of cuboid A is "<<cuboidA.surfaceArea()<<" units squared\n";
        //Call the member function volume() of cuboidB to find it's volume
        cout <<"Volume of cuboid B is "<<cuboidB.volume()<<" units cubed\n";
        //Call the member function lenSpaceDiag() of cuboidC to find the length of it's space
diagonal
        cout <<"Length of the space diagonal of cuboid C is "<<cuboidC.lenSpaceDiag()<<" units\n";
        return 0;
}

/*This program produced the following output when compiled using the GNU GCC C++ compiler from
within the Code::Blocks IDE running on an Ubuntu GNU/Linux operating system, by Robin Broad,
Newcastle upon Tyne, UK, May 2018:

C++ Classes and Objects Demo
Robin Broad
May 2018
Constructing a cuboid object of dimensions 4 x 5 x 2 units
Constructing a cuboid object of dimensions 8 x 12 x 4 units
Constructing a cuboid object of dimensions 5 x 10 x 3 units
Surface area of cuboid A is 76 units squared
Volume of cuboid B is 384 units cubed
Length of the space diagonal of cuboid C is 11.5758 units
Destructing a cuboid object of dimensions 5 x 10 x 3 units
Destructing a cuboid object of dimensions 8 x 12 x 4 units
Destructing a cuboid object of dimensions 4 x 5 x 2 units

Process returned 0 (0x0)   execution time : 0.006 s
Press ENTER to continue.
*/
```

# Section Three

## C++ Class Inheritance, Polymorphism and Friend Functions

## C++ Class Inheritance, Polymorphism and Friend Functions

This program demonstrates class inheritance, polymorphism and friend functions

- Inheritance - The subclasses Plastic, Metal and Ceramic are all materials and inherit some common properties from their parent (base) class Material

- Polymorphism - From the Greek meaning "many forms" this is an important principle of Object Orientated Programming. The derived classes can exhibit different forms of behaviour depending upon their properties. In this case, the properties member function is overridden in the derived classes. Also, the friend function twoPoundsOff(Material *poA) is able to operate on objects of the base class or derived classes, another example of Polymorphism

- Friend function - The class member cost is protected in the Material base class, the function twoPoundsOff(Material *poA)is a friend, it is authorised to change the class member

- Abstraction - This is the concept of simplifying real world things and representing their essential characteristics in software. For example, the plastic toy is represented as a plastic object with the properties of plastic, a density and a cost.

- Classification - The materials plastic, metal and ceramic are grouped together as members of the class material

The following code listing is heavily commented to give students a step by step explanation for every line of code and every new concept. The following code is a complete working C++ program which should compile and run without error.

This program is free software published under GNU GPL and can be downloaded free from http://www.robinbroad.co.uk/

```
/* C++ Class Inheritance, Polymorphism and Friend Demo
Copyright © Robin Broad 2018
Free software published under GNU GPL
http://www.robinbroad.co.uk/
May 2018
*/

//Preprocessor instructions to the C++ compiler
//Include the standard C input/ output library
#include <iostream>
//Use the standard names in namespace std for the functions in the iostream library (e.g. cout).
Without this, the compiler would not recognise the function cout, which we are using in this program
using namespace std;

//Base class
class Material
/*Class name Material. This class is designed to be a base class for the subclasses Plastic, Metal
and Ceramic. These subclasses are all materials and inherit some common properties from their parent
(base) class Material.
This code is heavily commented to assist students of C++
Author: Robin Broad
C++ Examples
http://www.robinbroad.co.uk/
May 2018*/
{
        protected:
        /*To make these class members accessible to subclasses (derived classes) these variables
need to be declared as protected, not private. (This is also known as data hiding. This is part of
the process of encapsulation; information hiding, data and methods)*/
        float density;
        //Material density in Kg/m^3
        float cost;
        //Object cost in UK £
        public:
        //***Constructor ***
        /*The constructor is a member function with the same name as the class.
        It is called whenever an object (class instance) is created*/
        Material(float fpDensity, float fpCost){
        //fp means function parameter etc.
                density=fpDensity;
                cost=fpCost;
        }

        void properties()
        /*Member function to display the general properties of the base class Material
        This member function is overridden by the derived classes Plastic, Metal and Ceramic
        Void means that this function does not return a value*/
        {
                cout << "I am a material. I am a solid and can be used in
manufacturing.\n"<<"Density: "<<density<<"Kg/m^3\nCost: £"<<cost << "\n\n";
        }

        /*Declaring the friend function twoPoundsOff
        This function deducts £2 from the cost of the object because it is being discounted for a
sale
        The class member cost is protected in this class, but because this function is a friend, it
is authorised to change the class member
        The function twoPoundsOff is not a member of the Material class
        We are passing a pointer to the Material object (rather than a copy) so that it's instance
member can be modified
        Void means that this function does not return a value*/
        friend void twoPoundsOff(Material *poA);
};
//}; marks the end of the class definition
```

```
/*First derived class
Plastic is a type of material*/
class Plastic : public Material
/*Class name Plastic. This class is derived from the parent (base) class Material. It inherits some of the properties of the base class Material.*/
{
        public:
        //***Constructor ***
        /*The constructor is a member function with the same name as the class.
        It is called whenever an object (class instance) is created
        This constructor is inherited from the base class*/
        Plastic(float a, float b):Material(a,b){}

        /*Function overriding
        The definition of the properties() function in the base class is overridden here*/
        void properties()
        {
                cout << "I am a plastic object. I am strong and hard wearing, I can be moulded and I am and electrical and heat insulator.\n"<<"Density: "<<density<<"Kg/m^3\nCost: £"<<cost << "\n\n";
        }

};
//}; marks the end of the class definition

/*Second derived class
metal is a type of material*/
class Metal : public Material
/*Class name Metal. This class is derived from the parent (base) class Material. It inherits some of the properties of the base class Material.*/
{
        public:
        //***Constructor ***
        /*The constructor is a member function with the same name as the class.
        It is called whenever an object (class instance) is created
        This constructor is inherited from the base class*/
        Metal(float a, float b):Material(a,b){}

        /*Function overriding
        The definition of the properties() function in the base class is overridden here*/
        void properties()
        {
                cout << "I am a metal object. I am strong, hard and shiny, and I am a good conductor of heat and electricity.\n"<<"Density: "<<density<<"Kg/m^3\nCost: £"<<cost << "\n\n";
        }

};
//}; marks the end of the class definition
```

```
/*Third derived class
Ceramic is a type of material*/
class Ceramic : public Material
/*Class name Ceramic. This class is derived from the parent (base) class Material. It inherits some of the properties of the base class Material.*/
{
        public:
        //***Constructor ***
        /*The constructor is a member function with the same name as the class.
        It is called whenever an object (class instance) is created
        This constructor is inherited from the base class*/
        Ceramic(float a, float b):Material(a,b){}

        /*Function overriding
        The definition of the properties() function in the base class is overridden here*/
        void properties()
        {
                cout << "I am a ceramic object. I am hard wearing but brittle, heat resistant and I am an electrical and heat insulator.\n"<<"Density: "<<density<<"Kg/m^3\nCost: £"<<cost << "\n\n";
        }

};
//}; marks the end of the class definition

/*Base class friend function
Applies a two pound discount to the object
This function is not a member of the class Material*/
void twoPoundsOff(Material *poA)
//We are passing a pointer to the Material object (rather than a copy) so that it's instance member can be modified*/
{
        (*poA).cost-=2;
        //(*poA) means the object that is pointed to by *poA
        //An alternative syntax is poA->cost-=2
        //x-=2 means x=x-2
}
/*We could have passed the object by reference, rather than using a pointer.
The syntax is as follows:
void twoPoundsOff(Material &poA){ poA.cost-=2;}
{Where & is the reference operator)
This time the member function would be called as follows:
twoPoundsOff(objectName); instead of twoPoundsOff(&objectName);
*/
```

```
//*** Main program ***
int main()
{
        //Output Header
        cout <<"C++ Class Inheritance, Polymorphism and Friend Demo\nRobin Broad\nMay 2018\n";
        /*This program demonstrates class inheritance, polymorphism and friend functions
        Inheritance - The subclasses Plastic, Metal and Ceramic are all materials and inherit some common properties from their parent (base) class Material
        Polymorphism - From the Greek meaning "many forms" this is an important principle of Object Orientated Programming. The derived classes can exhibit different forms of behaviour depending upon their properties. In this case, the properties member function is overridden in the derived classes. Also, the friend function twoPoundsOff(Material *poA) is able to operate on objects of the base class or derived classes, another example of Polymorphism
        Friend function - The class member cost is protected in the Material base class, the function twoPoundsOff(Material *poA)is a friend, it is authorised to change the class member
        Abstraction - This is the concept of simplifying real world things and representing their essential characteristics in software. For example, the plastic toy is represented as a plastic object with the properties of plastic, a density and a cost.
        Classification - The materials plastic, metal and ceramic are grouped together as members of the class material*/

        /*Create an object of the base class Material
        This wooden table is made from pine (density 420Kg/m^3) and costs £49.99*/
        Material woodenTable(420,49.99);

        /*Create an object of the derived class Plastic
        This toy is made from plastic (density 920Kg/m^3) and costs £3.50*/
        Plastic plasticToy(920,3.50);

        /*Create an object of the derived class Metal
        This saucepan is made from stainless steel (density 8000Kg/m^3) and costs £7.50*/
        Metal saucepan(8000,7.50);

        /*Create an object of the derived class Ceramic
        This teacup is made from ceramic (density 2130Kg/m^3) and costs £2.25*/
        Ceramic teaCup(2130,2.25);

        /*Display the properties of the wooden table
        The base class (Material) properties function is called*/
        cout<<"Properties of the wooden table: ";
        woodenTable.properties();

        /*Display the properties of the plastic toy
        The derived class Plastic overrides the properties function in the base class Material
        The plastic class properties function is called*/
        cout<<"Properties of the plastic toy: ";
        plasticToy.properties();

        /*Display the properties of the saucepan
        The derived class Metal overrides the properties function in the base class Material
        The Metal class properties function is called*/
        cout<<"Properties of the saucepan: ";
        saucepan.properties();

        /*Display the properties of the teacup
        The derived class Ceramic overrides the properties function in the base class Material
        The Ceramic class properties function is called*/
        cout<<"Properties of the teacup: ";
        teaCup.properties();
        //Discount the wooden table using a friend function
        twoPoundsOff(&woodenTable);
        //&woodenTable means the address of the object woodenTable
        cout<<"Properties of the discounted wooden table: ";
        woodenTable.properties();

        //Discount the plastic toy using a friend function
        twoPoundsOff(&plasticToy);
        //&plasticToy means the address of the object plasticToy
        cout<<"Properties of the discounted plastic toy: ";
        plasticToy.properties();

        return 0;
}

/*This program produced the following output when compiled using the GNU GCC C++ compiler from
within the Code::Blocks IDE running on an Ubuntu GNU/Linux operating system, by Robin Broad,
Newcastle upon Tyne, UK, May 2018:

C++ Class Inheritance, Polymorphism and Friend Demo
Robin Broad
May 2018
Properties of the wooden table: I am a material. I am a solid and can be used in manufacturing.
Density: 420Kg/m^3
Cost: £49.99

Properties of the plastic toy: I am a plastic object. I am strong and hard wearing, I can be moulded
and I am and electrical and heat insulator.
Density: 920Kg/m^3
Cost: £3.5

Properties of the saucepan: I am a metal object. I am strong, hard and shiny, and I am and good
conductor of heat and electricity.
Density: 8000Kg/m^3
Cost: £7.5

Properties of the teacup: I am a ceramic object. I am hard wearing but brittle, heat resistant and I
am an electrical and heat insulator.
Density: 2130Kg/m^3
Cost: £2.25

Properties of the discounted wooden table: I am a material. I am a solid and can be used in
manufacturing.
Density: 420Kg/m^3
Cost: £47.99

Properties of the discounted plastic toy: I am a plastic object. I am strong and hard wearing, I can
be moulded and I am and electrical and heat insulator.
Density: 920Kg/m^3
Cost: £1.5


Process returned 0 (0x0)   execution time : 0.007 s
Press ENTER to continue.
*/
```

```
/*
        Copyright © Robin Broad 2018
        This program is free software: you can redistribute it and/or modify
        it under the terms of the GNU General Public License as published by
        the Free Software Foundation, either version 3 of the License, or
        (at your option) any later version.

        This program is distributed in the hope that it will be useful,
        but WITHOUT ANY WARRANTY; without even the implied warranty of
        MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
        GNU General Public License for more details.

        GNU General Public License: https://www.gnu.org/licenses/gpl.html
*/
```

# **Section Four**
## C++ Arrays and Pointers

## C++ Arrays and Pointers

This program demonstrates passing arrays to functions by reference, passing pointers to arrays to functions and changing the contents of a memory address using a pointer.

The following code listing is heavily commented to give students a step by step explanation for every line of code and every new concept. The following code is a complete working C++ program which should compile and run without error.

This program is free software published under GNU GPL and can be downloaded free from http://www.robinbroad.co.uk/

```
/* C++ Arrays and Pointers Demo
Copyright © Robin Broad 2018
Free software published under GNU GPL
http://www.robinbroad.co.uk/
May 2018
*/

//Preprocessor instructions to the C++ compiler
//Include the standard C input/ output library
/*This code is heavily commented to assist students of C++
Author: Robin Broad
May 2018
*/
#include <iostream>
//Use the standard names in namespace std for the functions in the iostream library (e.g. cout). Without this, the compiler would not recognise the function cout, which we are using in this program
using namespace std;
//Include the standard C maths library. We need it to calculate the square [pow(x,2)] of the array elements
#include<math.h>

//*** Defining Functions ***
void printArray(int passedArrayReference[], int arraySize){
        //This function will print out all of the elements of the array passed to it
        //The array is passed by reference to the function. This means that the function does not receive a copy of the array, but a reference to it. Other arguments are passed by value and cannot be altered in the main program by the function.
        //Void means that this function does not return a value
        for (int i = 0; i < arraySize; ++i) {
                cout << passedArrayReference[i];
                if (i< arraySize-1) cout << ", ";
                //We don't need a comma after the last number printed
        }
        cout <<"\n";
}

void fillArrayWithSequence(int passedArrayReference[], int arraySize){
        //This function will fill the array with a sequence of integers starting from 1
        //The array is passed by reference to the function. This means that the function does not receive a copy of the array, but a reference to it. Other arguments are passed by value and cannot be altered in the main program by the function.
        //Void means that this function does not return a value
        for (int i = 0; i < arraySize; ++i) {
                passedArrayReference[i]=i+1;
                //+1 because we want to start at 1, not 0
        }
}
```

```cpp
void squareArrayElements(int passedArrayReference[], int arraySize){
        //This function will square the elements of the array
        //The array is passed by reference to the function. This means that the function does not receive a copy of the array, but a reference to it. Other arguments are passed by value and cannot be altered in the main program by the function.
        //Void means that this function does not return a value
        for (int i = 0; i < arraySize; ++i) {
                passedArrayReference[i] = pow(passedArrayReference[i],2);
                //pow(x,2) means calculate the square (power of 2) of the number
        }
}

void arrayPointerEdit(int *passedArrayPointer, int arraySize){
        //Squaring the elements in the array again, this time using a pointer
        //Void means that this function does not return a value
        for (int i = 0; i < arraySize; ++i) {
                cout << "Overwriting contents of memory address: " << passedArrayPointer <<"\n";
                *passedArrayPointer= pow(*passedArrayPointer,2);
                //pow(x,2) means calculate the square (power of 2) of the number
                passedArrayPointer++;
        }
}
```

```
//*** Main program ***
int main() {

        //Output Header
        cout <<"C++ Arrays and Pointers Demo\nRobin Broad\nMay 2018\n";
        /*This program demonstrates passing arrays to functions by reference, passing pointers to arrays to functions and changing the contents of a memory address using a pointer*/

        //Declaring variables
        int arraySize=10;
        int arrayA[arraySize] = {0};
        //This declares the array as an array of integers and initialises all of the elements of the array to zero

        //Declaring pointers
        int *pointerToArrayA=arrayA;
        //The * is the dereference operator which is used to declare that this is a pointer. It is assigned to hold the start address of the array. It's type should also match the type of the data that it points to.
        int *pointerToArraySize=&arraySize;
        //The * is the dereference operator which is used to declare that this is a pointer. It is assigned to hold the memory address of the variable arraySize. It must have the same type as the data it points to.
        //The & is the reference operator which gives us the memory address of the variable arraySize

        //Call a function to print the array
        cout << "Zero initialised array:\n";
        printArray(arrayA,arraySize);

        //Call a function to fill the array with a sequence of integers starting from 1
        fillArrayWithSequence(arrayA,arraySize);
        cout << "Integer sequence array:\n";
        printArray(arrayA,arraySize);

        //Call a function to square the elements of the array
        squareArrayElements(arrayA,arraySize);
        cout << "Squared elements array:\n";
        printArray(arrayA,arraySize);

        //Call a function to square the elements in the array again, this time using a pointer
        cout << "Squaring the elements in the array again, this time using a pointer:\n";
        arrayPointerEdit(pointerToArrayA,arraySize);
        cout << "Reprinting the array:\n";
        printArray(arrayA,arraySize);

        //Manipulate the array size by changing the contents of a memory address
        cout << "The arraySize variable is stored at memory address: " << pointerToArraySize << " and holds the value " << *pointerToArraySize <<"\n";
        cout << "Manipulating contents of memory address: " << pointerToArraySize<<"\n";
        *pointerToArraySize=5;
        cout << "Contents of memory address: " << pointerToArraySize << " is now " << *pointerToArraySize<<"\n";
        cout << "Printing the array again, just to check:\n";
        printArray(arrayA,arraySize);
        return 0;
}
```

```
/*This program produced the following output when compiled using the GNU GCC C++ compiler from
within the Code::Blocks IDE running on an Ubuntu GNU/Linux operating system, by Robin Broad,
Newcastle upon Tyne, UK, May 2018:

C++ Arrays and Pointers Demo
Robin Broad
May 2018
Zero initialised array:
0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Integer sequence array:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Squared elements array:
1, 4, 9, 16, 25, 36, 49, 64, 81, 100
Squaring the elements in the array again, this time using a pointer:
Overwriting contents of memory address: 0xbfefe7d0
Overwriting contents of memory address: 0xbfefe7d4
Overwriting contents of memory address: 0xbfefe7d8
Overwriting contents of memory address: 0xbfefe7dc
Overwriting contents of memory address: 0xbfefe7e0
Overwriting contents of memory address: 0xbfefe7e4
Overwriting contents of memory address: 0xbfefe7e8
Overwriting contents of memory address: 0xbfefe7ec
Overwriting contents of memory address: 0xbfefe7f0
Overwriting contents of memory address: 0xbfefe7f4
Reprinting the array:
1, 16, 81, 256, 625, 1296, 2401, 4096, 6561, 10000
The arraySize variable is stored at memory address: 0xbfefe808 and holds the value 10
Manipulating contents of memory address: 0xbfefe808
Contents of memory address: 0xbfefe808 is now 5
Printing the array again, just to check:
1, 16, 81, 256, 625

Process returned 0 (0x0)   execution time : 0.015 s
Press ENTER to continue.
*/

/*
        Copyright © Robin Broad 2018
        This program is free software: you can redistribute it and/or modify
        it under the terms of the GNU General Public License as published by
        the Free Software Foundation, either version 3 of the License, or
        (at your option) any later version.

        This program is distributed in the hope that it will be useful,
        but WITHOUT ANY WARRANTY; without even the implied warranty of
        MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
        GNU General Public License for more details.

        GNU General Public License: https://www.gnu.org/licenses/gpl.html
*/
```

# Section Five

## Compiling and Running C++ Programs

## Compiling and Running C++ Programs

C++ programs can be written using a simple text editor:

- Linux - **gedit, Vim, Emacs**
- MacOS - **TextEdit, TextMate, Atom, Aquamacs**
- Windows - **Notepad, Notepad ++, Atom**

All of the programs in this guide were written using the gedit text editor in Linux. This text editor has the advantage of recognising C++ code and formatting it in colour (syntax highlighting).

C++ files are saved with the **file extension .cpp**
For example:
ClassesAndObjects.cpp

A compiler converts the C++ program into executable machine code. The **GNU C++ compiler** is freely available from the **Free Software Foundation**:
https://gcc.gnu.org/

The programs in this guide were compiled using the **Code::Blocks IDE** (integrated development environment). Code::Blocks is a free cross-platform IDE which runs on Linux, Mac and Windows.
http://www.codeblocks.org/

**MinGW** (Minimalist GNU for Windows) is a free programming environment for Windows.
http://www.mingw.org/

The **Nanyang Technological University** in Singapore have produced an excellent C++ introduction here:

https://www.ntu.edu.sg/home/ehchua/programming/cpp/cp0_Introduction.html

## **Section Six**
## GNU Free Documentation License

## GNU Free Documentation License

Version 1.3, 3 November 2008

### 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

### 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

### 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

### 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

### 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

### 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

### 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

### 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

### 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

### 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

### 11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A

public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.
"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.
"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.
An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.
The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

## ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.