

**PROGRAMMING IN C++**

**CST-153**

SEMESTER-2

**Submitted To: Submitted By:** Er.Shallini Anchal Malik

16BCS1828

Mrmmsm

**TITLE**

C++ Program to Optimize wire Length

in Electrical circuit.

**ABOUT THE PROJECT**

In this program , we are running a code for optimizing the wire length in electric circuit.

**Electrical wiring** is commonly understood to be an [electrical](https://en.wikipedia.org/wiki/Electric_power) installation for operation by end users within a building, an engineered structure or a designated outdoor location. It includes the cabling and associated devices such as switches, distribution boards, sockets and light fittings but does not cover the [transmission of electrical power](https://en.wikipedia.org/wiki/Electrical_power_transmission) to the installation.

Such wiring is subject to rigorous safety standards for design and installation. Allowable [Wires](https://en.wikipedia.org/wiki/Wire) and [cables](https://en.wikipedia.org/wiki/Cables) types and sizes are specified according to the circuit operating [voltage](https://en.wikipedia.org/wiki/Voltage) and [electric current](https://en.wikipedia.org/wiki/Electric_current) capability, with further restrictions on the environmental conditions, such as ambient temperature range, moisture levels, and exposure to sunlight and chemicals. Associated circuit protection, control and distribution devices within a building's wiring system are subject to voltage, current and functional specification.

To ensure both wiring and associated devices are designed, selected and installed so that they are safe for use, they are subject to wiring safety codes or regulations, which vary by locality, country or region. The [International Electrotechnical Commission](https://en.wikipedia.org/wiki/International_Electrotechnical_Commission) (IEC) is attempting to harmonise wiring standards amongst member countries, but significant variations in design and installation requirements still exist.

This is a C++ Program to optimize wire length in electic circuit. This problem can be reduced to finding the shortest path between two components

Here is source code of the C++ Program to Optimize Wire Length in Electrical Circuit. The C++ program is successfully compiled and run on a Linux system. The program output is also shown below.

**ABOUT THE LANGUAGE –C++**

C++ is a [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). It has [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) and [generic](https://en.wikipedia.org/wiki/Generic_programming) programming features, while also providing facilities for [low-level](https://en.wikipedia.org/wiki/Low-level_programming) [memory](https://en.wikipedia.org/wiki/Memory_(computing)) manipulation.

It was designed with a bias toward [system programming](https://en.wikipedia.org/wiki/System_programming) and [embedded](https://en.wikipedia.org/wiki/Embedded_software), resource-constrained and large systems, with [performance](https://en.wikipedia.org/wiki/Performance_(software)), efficiency and flexibility of use as its design highlights.[[5]](https://en.wikipedia.org/wiki/C%2B%2B#cite_note-Stroustrup1-5) C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications,[[5]](https://en.wikipedia.org/wiki/C%2B%2B#cite_note-Stroustrup1-5) including [desktop applications](https://en.wikipedia.org/wiki/Application_software), servers (e.g. [e-commerce](https://en.wikipedia.org/wiki/E-commerce), [web search](https://en.wikipedia.org/wiki/Web_search_engine) or [SQL](https://en.wikipedia.org/wiki/SQL) servers), and performance-critical applications (e.g. [telephone switches](https://en.wikipedia.org/wiki/Telephone_switches) or [space probes](https://en.wikipedia.org/wiki/Space_probes)).[[6]](https://en.wikipedia.org/wiki/C%2B%2B#cite_note-applications-6) C++ is a [compiled language](https://en.wikipedia.org/wiki/Compiled_language), with implementations of it available on many platforms. Many vendors provide [C++ compilers](https://en.wikipedia.org/wiki/List_of_compilers#C.2B.2B_compilers), including the [Free Software Foundation](https://en.wikipedia.org/wiki/Free_Software_Foundation), [Microsoft](https://en.wikipedia.org/wiki/Microsoft), [Intel](https://en.wikipedia.org/wiki/Intel), and [IBM](https://en.wikipedia.org/wiki/IBM).

C++ is standardized by the [International Organization for Standardization](https://en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO), with the latest standard version ratified and published by ISO in December 2014 as [*ISO/IEC 14882*](https://en.wikipedia.org/wiki/C%2B%2B#Standardization)*:2014* (informally known as [C++14](https://en.wikipedia.org/wiki/C%2B%2B14)).[[7]](https://en.wikipedia.org/wiki/C%2B%2B#cite_note-isocpp2011-7) The C++ programming language was initially standardized in 1998 as *ISO/IEC 14882:1998*, which was then amended by the [C++03](https://en.wikipedia.org/wiki/C%2B%2B03), *ISO/IEC 14882:2003*, standard. The current C++14 standard supersedes these and [C++11](https://en.wikipedia.org/wiki/C%2B%2B11), with [new features](https://en.wikipedia.org/wiki/C%2B%2B14) and an enlarged [standard library](https://en.wikipedia.org/wiki/C%2B%2B#Standard_library). Before the initial standardization in 1998, C++ was developed by [Bjarne Stroustrup](https://en.wikipedia.org/wiki/Bjarne_Stroustrup) at [Bell Labs](https://en.wikipedia.org/wiki/Bell_Labs) since 1979, as an extension of the [C language](https://en.wikipedia.org/wiki/C_(programming_language)) as he wanted an efficient and flexible language similar to C, which also provided high-level features for program organization. The [C++17](https://en.wikipedia.org/wiki/C%2B%2B17) standard is due in July 2017, with the draft largely implemented by some compilers already, and C++20 is the next planned standard thereafter.

**FEATURES OF C++**

* **Data Abstraction**

**Data abstraction** refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.

* **Data encapsulation**

**Encapsulation** is an Object Oriented Programming concept that binds together the**data** and functions that manipulate the **data**, and that keeps both safe from outside interference and misuse. ... **C++** supports the properties of **encapsulation** and **data**hiding through the creation of user-defined types, called classes.

* **Data Hiding**

**Data hiding** is a software development technique specifically used in object-oriented programming (OOP) to **hide** internal object details (**data** members). **Data hiding**ensures exclusive **data** access to class members and protects object integrity by preventing unintended or intended changes.

* **Message Passing**

**Message Passing**. An object-oriented program consists of a set of objects that communicate with each other.

* **Dynamic Binding**

**Dynamic binding** also called **dynamic** dispatch is the process of linking procedure call to a specific sequence of code (method) at run-time. ... **Dynamic binding** is also known as late **binding** or run-time **binding**. **Dynamic binding** is an object oriented programming concept and it is related with polymorphism and inheritance.

* **Polymorphism**

The word **polymorphism** means having many forms. Typically, **polymorphism**occurs when there is a hierarchy of classes and they are related by inheritance. **C++ polymorphism** means that a call to a member function will cause a different function to be executed depending on the type of object that invokes the function.

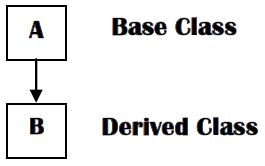
* **Inheritance**

**Inheritance** in Object Oriented Programming can be described as a process of creating new classes from existing classes. New classes **inherit** some of the properties and behavior of the existing classes. An existing class that is "parent" of a new class is called a base class.

**TYPES OF INHERITANCE**

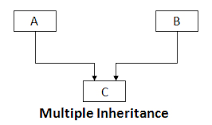
* **SINGLE LEVEL**

**When one class is derived from another class, it is referred to as single level inheritance.**



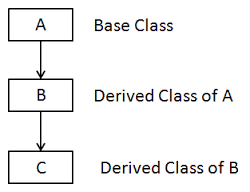
* **MULTIPLE**

**When a single class is derived from more than one bases classes, it is called multiple inheritance.**

****

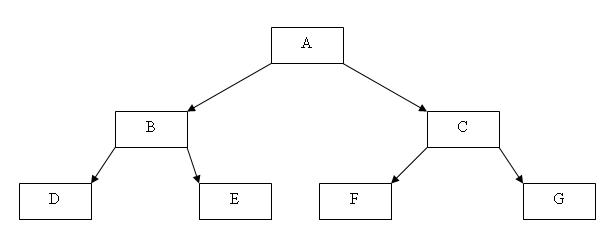
* **MULTILEVEL**

**When one class is derived from another class and one more class is derived from the previously derived class, it is then called multilevel inheritance.**

****

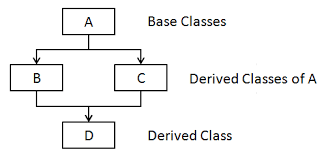
* **Hierarchical**

**When more than one class is derived from a single base class, it is called hierarchical inheritance.**

****

* **Hybrid**

**When all types of inheritance are used to derive a class, it is called Hybrid inheritance.**

****

**ADVANTAGES OF INHERITANCE**

- It allows the code to be reused as many times as needed. 

- The base class once defined and once it is compiled, it need not be reworked.

- Saves time and effort as the main code need not be written again.

- Code reusability

- Saves time in program development.

**TEMPLATES IN C++**

**Templates** are a feature of the C++ programming language that allows functions and classes to operate with generic types. This allows a function or class to work on many different data types without being rewritten for each one.

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

There is a single definition of each container, such as **vector**, but we can define many different kinds of vectors for example, **vector <int>** or **vector <string>**.

**ADVANTAGES OF TEMPLATES**

* reducing the repetition of code (generic containers, algorithms)
* reducing the repetition of code advanced (MPL and Fusion)
* static polymorphism (=performance) and other compile time calculations
* policy based design (flexibility, reusability, easier changes, etc)
* increasing safety at no cost (i.e. dimension analysis via Boost Units, static assertions, concept checks)

**FILE HANDLING IN C++**

**File Handling** concept in C++ language is used for store a data permanently in

computer. Using file handling we can store our data in Secondary memory (Hard disk).

**Why use File Handling?**

* For permanent storage.
* The transfer of input - data or output - data from one computer to another can be easily done by using files.

**How to achieve File Handling?**

For achieving file handling in C++ we need follow following steps

* Naming a file
* Opening a file
* Reading data from file
* Writing data into file
* Closing a file

**SOURCE CODE**

#include <stdio.h>

#include <limits.h>

#include <iostream>

using namespace std;

#define V 9

int minDistance(int dist[], bool sptSet[])

{

int min = INT\_MAX, min\_index;

for (int v = 0; v < V; v++)

if (sptSet[v] == false && dist[v] <= min)

min = dist[v], min\_index = v;

return min\_index;

}

void printSolution(int dist[], int n)

{

cout << "Component\tDistance from other component\n";

for (int i = 0; i < V; i++)

printf("%d\t\t%d\n", i, dist[i]);

}

void optimizeLength(int graph[V][V], int src)

{

int dist[V];

bool sptSet[V];

for (int i = 0; i < V; i++)

dist[i] = INT\_MAX, sptSet[i] = false;

dist[src] = 0;

for (int count = 0; count < V - 1; count++)

{

int u = minDistance(dist, sptSet);

sptSet[u] = true;

for (int v = 0; v < V; v++)

if (!sptSet[v] && graph[u][v] && dist[u] != INT\_MAX && dist[u]

+ graph[u][v] < dist[v])

dist[v] = dist[u] + graph[u][v];

}

printSolution(dist, V);

}

int main()

{

int graph[V][V] =

{ { 0, 4, 0, 0, 0, 0, 0, 8, 0 }, { 4, 0, 8, 0, 0, 0, 0, 11, 0 }, {

0, 8, 0, 7, 0, 4, 0, 0, 2 },

{ 0, 0, 7, 0, 9, 14, 0, 0, 0 }, { 0, 0, 0, 9, 0, 10, 0, 0,

0 }, { 0, 0, 4, 0, 10, 0, 2, 0, 0 }, { 0, 0, 0, 14,

0, 2, 0, 1, 6 }, { 8, 11, 0, 0, 0, 0, 1, 0, 7 }, {

0, 0, 2, 0, 0, 0, 6, 7, 0 } };

cout << "Enter the starting component: ";

int s;

cin >> s;

optimizeLength(graph, s);

return 0;

}

**OUTPUT**

Enter the starting component: 2

Component Distance from other component

0 12

1 8

2 0

3 7

4 14

5 4

6 6

7 7

8 2

**ENHANCEMENTS THAT CAN BE DONE IN THE PROJECT**

* **The code can be done using file handling in order to store the data and information permanently.**
* **The code can also be done using classes and objects for efficient programming.**
* **There can also be the use of pointers in order to reduce the memory space occupied by the variables.**

**REFERENCES**

* **Programming in c++ by Sumita Arora.**
* [**www.wikipedia.com**](http://www.wikipedia.com)
* [**www.sanfoundry.com**](http://www.sanfoundry.com)
* [**www.tutorialspoint.com**](http://www.tutorialspoint.com)
* [**www.geeksforgeeks.com**](http://www.geeksforgeeks.com)