**C++ General Overview**

C++ is a general-purpose [programming language](https://en.wikipedia.org/wiki/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 memory manipulation.

It is designed with a bias toward [system programming](https://en.wikipedia.org/wiki/System_programming) and embedded, resource-constrained and large systems, with performance, efficiency and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, 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), performance-critical applications (e.g. [telephone switches](https://en.wikipedia.org/wiki/Telephone_switches) or [space probes](https://en.wikipedia.org/wiki/Space_probes)), and entertainment software. C++ is a [compiled](https://en.wikipedia.org/wiki/Compiled_language) language, with implementations of it available on many platforms.

C++ is standardized by the [International Organization for Standardization](https://en.wikipedia.org/wiki/International_Organization_for_Standardization). C++ was developed by [Bjarne Stroustrup](https://en.wikipedia.org/wiki/Bjarne_Stroustrup) at [Bell Labs](https://en.wikipedia.org/wiki/Bell_Labs) in 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.

Many other programming languages have been influenced by C++, including [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), and newer versions of C.

#### Object Oriented Programming Concepts

#### Encapsulation

[Encapsulation](https://en.wikipedia.org/wiki/Information_hiding) is the hiding of information to ensure that data structures and operators are used as intended and to make the usage model more obvious to the developer. C++ provides the ability to define classes and functions as its primary encapsulation mechanisms. Within a class, members can be declared as either public, protected, or private to explicitly enforce encapsulation.  
 A public member of the class is accessible to any function. A private member is accessible only to functions that are members of that class and to functions and classes explicitly granted access permission by the class ("friends"). A protected member is accessible to members of classes that inherit from the class in addition to the class itself and any friends.

It is generally considered good practice to make all [data](https://en.wikipedia.org/wiki/Data) private or protected, and to make public only those functions that are part of a minimal interface for users of the class. This can hide the details of data implementation, allowing the designer to later fundamentally change the implementation without changing the interface in any way.

#### Inheritance

[Inheritance](https://en.wikipedia.org/wiki/Inheritance_(computer_science)) allows one data type to acquire properties of other data types. Inheritance from a [base class](https://en.wikipedia.org/wiki/Base_class) may be declared as public, protected, or private. This access specifier determines whether unrelated and derived classes can access the inherited public and protected members of the base class. Only public inheritance corresponds to what is usually meant by "inheritance". If the access specifier is omitted, a "class" inherits privately, while a "struct" inherits publicly.  
 Base classes may be declared as virtual; this is called [virtual inheritance](https://en.wikipedia.org/wiki/Virtual_inheritance). Virtual inheritance ensures that only one instance of a base class exists in the inheritance graph, avoiding some of the ambiguity problems of multiple inheritance.

[Multiple inheritance](https://en.wikipedia.org/wiki/Multiple_inheritance) is a C++ feature not found in most other languages, allowing a class to be derived from more than one base class; this allows for more elaborate inheritance relationships. For example, a "Flying Cat" class can inherit from both "Cat" and "Flying Mammal". A class that does not have any derived classes or functions, i.e it is a pure base class is called an abstract class. The member functions of such an abstract base class are normally explicitly defined in the derived class, not inherited implicitly.

### Polymorphism

[Polymorphism](https://en.wikipedia.org/wiki/Type_polymorphism) enables one common interface for many implementations, and for objects to act differently under different circumstances.

C++ supports several kinds of *static* ([compile-time](https://en.wikipedia.org/wiki/Compile-time)) and *dynamic* ([run-time](https://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase))) [polymorphisms](https://en.wikipedia.org/wiki/Polymorphism_(computer_science)), supported by the language features described above.

#### Static polymorphism

Function Overloading

[Function overloading](https://en.wikipedia.org/wiki/Function_overloading) allows programs to declare multiple functions having the same name but with different arguments (i.e. [*ad hoc* polymorphism](https://en.wikipedia.org/wiki/Ad_hoc_polymorphism)). The functions are distinguished by the number or types of their [formal parameters](https://en.wikipedia.org/wiki/Parameter_(computer_science)). Thus, the same function name can refer to different functions depending on the context in which it is used. The type returned by the function is not used to distinguish overloaded functions and would result in a compile-time error message.

#### Dynamic polymorphism

##### Inheritance

The feature in C++ by which a class can derive or inherit the properties and characteristics of a base class, including functions, variables, etc. is called inheritance.

**PROJECT DESCRIPTION**

The project is billing software of a large mall titled SWARTZ MART- it contains a large variety and number of products. It is developed using C++ and its various concepts including File Concept, Class and Structures, etc.

The billing software contains 127 pre-defined products with product codes as follows:  
a) Groceries(101-140)  
b) Cookery(201-220)  
c) Accessories(301-325)  
d) Electronics(401-415)  
e) Snacks(501-517)  
f) Miscellaneous(601-610)

The billing software’s highlighting features are:-

1. Easy and intuitive User Interface.
2. Command Line
3. Products from various code ranges and types, including groceries, electronics, accessories, etc.
4. Drop-down menus
5. Admin Username and Password only can be used to access certain sections of the software
6. ZPoints for additional benefits
7. Easy management of product and customer database
8. Scroll Bars(using arrow keys)

**PROJECT MEMBER DETAILS-**

1. Nihesh Anderson- Software Designer
2. Nishant Sharma- Database Manager& Report
3. Mihir Sriram- Graphics Assist

**FUNCTIONS USED:-**

1. void polygon(int,int[],int,int);  
    Creates a polygon (no of vertices,array of points,style of box,color of box)
2. void cbc(char[],int,int);  
    shows cursor while accepting character by character
3. void createaccount();   
    Creates a salesman account
4. void nbn(int &,int,int);  
    Accepts number by number
5. void personalinfo(account);   
    Stores workers information
6. void pay();  
    To pay a bill
7. void homepage();  
    Main page of the software
8. void billing();   
    To bill products
9. void defaultdisplay();  
    Official SWARTZ MART display style
10. void membercreate();  
     Creates Zpoints member
11. void viewstock();   
    To view the stock availability of a product
12. void viewdatabase();  
     To access the internal database of the shop
13. void stockupdate();   
    Updates the stock once a product is billed
14. void editstock();  
     To increase stock on arrival
15. void createproduct();  
     To add new products to the database
16. void clearslot(int,int);   
    clears a blank in a position
17. void deleteproduct();   
    Deletes a product from the database
18. void editproduct();  
     To edit a product in the database
19. void background(int);  
     To apply background effects
20. void workers();  
     To display the list of workers in the shop
21. void datetime();  
     Shows date and time
22. void zupdate();  
     Updates Zpoints after billing
23. void motion();  
     Welcome screen
24. void zpay();   
    Payment via Zpoints portal
25. void helpprint(int);  
     Default help menu style
26. void text();  
     Part of motion
27. void b\_three();  
     Part of motion
28. void dotmotion();  
     Part of motion
29. void b\_two();   
     Part of motion
30. void b\_one();   
     Part of motion
31. void showall();  
     Displays all Zpoints customers
32. void zhome();   
    Homepage for zpoints
33. void clearbill();   
    Clears the billing area in billing()
34. void clearmemory(int,int[],int[]);  
     Similar to a destructor. Reinitializes variable's value to null.
35. void decoder(char[],int,int);   
    Decodes the password
36. int authenticator(int,int);  
     Validates the password during login
37. void prodhelp();   
    Lists all products during billing
38. void dropdownbox(char[][100],int,int,int,char[]);  
     Creates a drop down box that gives options to the user
39. void sort(product[],int);  
     Internal function that sorts the different products into an array
40. void credits();   
     Credits page

**FILES USED**

The following files were used in the project for various purposes.

File names and use of the file are as follows:-

1. DATABASE FILES
   1. LIST.dat-  
      This file contains the database of all products categorized into various sub-types.
   2. ACCDET.dat  
      This contains the account details of the staff members, including the username and password.
   3. ZPOINTS.dat  
      A file containing details of the customers database including the ZPoints earned.
2. TEXT FILES
   1. CURBILL.txt  
      Stores the current bill number.
   2. CUSTDET.txt  
      Stores the details of the customer.
   3. DETAILS.txt  
      This contains the personal details of all staff members of SWARTZ MART.

**CONCLUSION**

The project has a very extensive scope of improvement. As it is based on a general, real life situation, it can be improved based on suggestions given by customers, the operators, etc.

New features like search by product code or name, cache memory for user login details, usage of drop down boxes at a higher level and more effectively, integration with Operating Systems and / or printers and data security can be implemented and thus the project can be improved.

**BIBLIOGRAPHY**

1. Computer Science with C++ by Sumita Arora
2. [www.wikipedia.org](http://www.wikipedia.org)
3. [www.stackoverflow.com](http://www.stackoverflow.com)
4. [www.cplusplus.org](http://www.cplusplus.org)
5. [www.borlandc.org](http://www.borlandc.org)