Experiment No.8 (CBS)

1. **Title:** Bank Account Management using structure.

# Problem Statement:

Write a program in C to define a structure for Customer bank account that holds Information like Account Number, Name of account holder, balance, Internet banking facility availed (Yes or No), Pin code (422001 to 422013), Account type (saving, recurring, deposit)

* 1. Read account details for n customers
  2. Identify the golden, silver and general customers.
     1. Golden customers: Balance> 10,00000
     2. Silver Customers: Balance >500000 and <10,00000
     3. General customers: Balance <500000
  3. Display the list of customers availing the Internet banking facility
  4. Display the customers belonging to a particular geographical location depending on postal code
  5. Display the customer list as per their account type

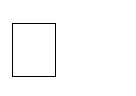
# Objectives:

* Students will able to understand user defined data types.
* Students will able to understand Concept of Structure & array of structure

# Outcomes:

* To understand various kinds of user defined datatypes.
* To understand how to implement the concepts structure and array structure.
* To implement the Bank Account structure for n number of account holder.

# Software/Hardware/Other Requirements:

* Any CPU with Pentium Processor / similar, 256 MB RAM/ more, 1GB HDD / more.
* Operating System – ubuntu/Fedora 64bit OS Software: G++ compiler/ GCC compiler, Code Editor

# Theory:

1. **Data Types in C++**

The data-type in a programming language is the collection of data with values having fixed meaning as well as characteristics. Some of them are an integer, floating point, character, etc. Usually, programming languages specify the range values for given data- type.

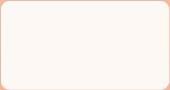
C++ Data Types are used to:

1. Identify the type of a variable when it declared.
2. Identify the type of the return value of a function.
3. Identify the type of a parameter expected by a function.



|  |
| --- |
|  |
|  |
|  |
|  |

1. User Defined Data Types:



C++ allows the feature called type definition which allows programmers to define their identifier that would represent an existing data type. There are 4 such types:

|  |  |
| --- | --- |
| Data Types | Description |
| Structure | It is a package of variables of different types under a single name. This is done to handle data efficiently. "struct" keyword is used to define a structure. |
| Union | These allow storing various data types in the same memory location. Programmers can define a union with different members, but only a single member can contain a value at given time. It is used for |
| Enum | Enumeration is a special data type that consists of integral constants, and  each of them is assigned with a specific name. "enum" keyword is used to define the enumerated data type. |
| Class | It is a package of variables of different types and member functions under a single name. This is done to handle data efficiently. "class" keyword is used to define a class. |

# Structure:

* Arrays allow to define type of variables that can hold several data items of the same kind. Similarly, structure is another user defined data type available in C that allows combining data items of different kinds. Structures are used to represent a record.
* A structure is a convenient tool for handling a group of logically related data items.
* For example, say that Book is group or collection of things such title, author, call number, publisher, no\_of\_pages, date\_of\_purchase, etc so as such data all items are of various type like author is string, no\_of\_pages is integer so we use structure for grouping such various types of items.
* Uses of Structures
  1. Database Management system like Employee in organization, books in library, items in store and soon...
  2. Changing the size of the cursor, Sending the output to printer
  3. Clearing the contents of the screen, Receiving a key from the keyboard
  4. Placing the cursor at an appropriate position on screen
  5. Drawing any graphics shape on the screen, Interacting with the mouse
* Defining Structure in C++:

The struct keyword is used to declare the structure in C++. Variables inside the structure are called members of the structure.

Syntax:

struct structureName

{

//member definitions

};

Example: struct Courses

{ char WebSite[50]; char Subject[50];

int HitCount;

};

Declaring & Initialization of Structure variable:

A structure variable declaration is similar to the declaration of any other data type. Also, we can initialize structure variable as same as array initialization.

Example:

struct Courses b1= [{“w](http://www.hulekuldeep.weebly.com/)ww[.hulekuldeep.weebly.com](http://www.hulekuldeep.weebly.com/)”, “OOP through C++”, 500}

When compiler comes across declaration statement, it reserves contiguous memory space for structure elements into memory as:

b1.WebSite b1.Subject b1.HitCount

|  |  |  |
| --- | --- | --- |
| [www.hulekuldeep.weebly.com](http://www.hulekuldeep.weebly.com/) | OOP through C++ | 500 |

100 150 200 204

Accessing Structure Members

To access any member of a structure, we use the member access operator (.). The member access operator is coded as a period between the structure variable name and the structure member that we wish to access.

Let’s see example for all these concepts through C++ -program:

|  |
| --- |
| #include<iostream> using namespace std; struct Courses  { char WebSite[50]; char Subject[50]; int Price;  };  void main( )  {  // Initialization  struct Courses C={“[www.hulekuldeep.weebly.com](http://www.hulekuldeep.weebly.com/)”, “OOP through C++”, 500};  //Print  cout<<"\nWebSite :"<<C.WebSite; cout<<"\nSubject  :"<<C.Subject; cout<<"\nWebSiteHitcount  :”<<C.HitCount; return 0;  } |
| Program output:  WebSite : [www.hulekuldeep.weebly.com](http://www.hulekuldeep.weebly.com/) Subject: OOP through C++  WebSite Hitcount:500 |

1. Array of Structure:

* Structure is collection of different data type. An object of structure represents a single record in memory, if we want more than one record of structure type, we have to create an array of structure or object.
* As we know, an array is a collection of similar type, therefore an array can be of structure type.

|  |  |
| --- | --- |
| Syntax for declaring structure array structstruct-name  {  datatype var1; datatype var2;  - - - - - - - - -  datatype varN;  };  structstruct-name obj [ size ]; | Example : structBookinfo  {  char[20] bname; int pages;  int price;  }Book[100]; |

* Explanation :

1. Here Book structure is used to Store the information of one Book.
2. In case if we need to store Information of 100 books then Array of Structure isused.
3. b1[0] stores the Information of 1st Book, b1[1] stores the information of 2nd Book and So on We can store the information of 100 books.
   * book [3] is shown Below
   * Accessing Pages field of Second Book: Book[1].pages

|  |
| --- |
| Live Example for Array of Structure |
| #include <stdio.h> struct Bookinfo {  char[20] bname; int pages;  int price;  }book[3]; int main() {  int i; for(i=0;i<3;i++) {  cout << "\nEnter the Name of Book: "; gets(book[i].bname);  cout<<"\nEnter the Number of Pages:"; cin>>book[i].pages;  cout<<"\nEnter the Price of Book : "; cin>>book[i].price;  }  cout<<"\n--------- Book Details ";  for(i=0;i<3;i++) {  cout<<"\nName of Book : %s",book[i].bname; cout<<"\nNumber of Pages : %d",book[i].pages; cout<<"\nPrice of Book : %f",book[i].price;  }  return 0;  } |

Program output:

Enter the Name of Book: ABC Enter the Number of Pages : 100 Enter the Price of Book : 200 Enter the Name of Book: EFG Enter the Number of Pages : 200 Enter the Price of Book : 300 Enter the Name of Book: HIJ Enter the Number of Pages : 300 Enter the Price of Book : 500

--------- Book Details ------------Name of

Book: ABC

Number of Pages : 100 Price of Book : 200 Name of Book: EFG Number of Pages : 200 Price of Book : 300 Name of Book: HIJ Number of Pages : 300 Price of Book : 500

1. Union

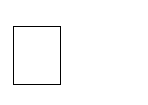
 A union is a special data type available in C/C++ that enables you to store different data types in the same memory location.

 You can define a union with many members, but only one member can contain a value at any given time.

 Unions provide an efficient way of using the same memory location for multipurpose.

Declaration of Union

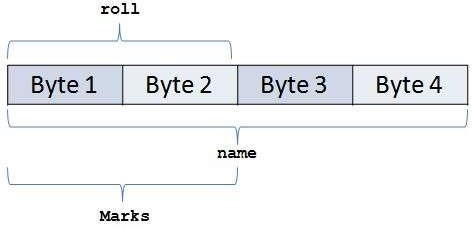
 Union is similar to that of Structure. Syntax of both are same but major difference between structure and union is ‘memory storage ‘.

 In structures, each member has its own storage location, whereas all the members of union use the same location. Union contains many members of different types,  Union can handle only one member at a time.

 Note: Unions are Declared in the same way as a Structure. Only “struct Keyword” is replaced with union.

|  |  |
| --- | --- |
| Syntax for declaring union union [union tag]  { member definition; member definition;  ...  member definition; } [one or more union variables]; | Example : |

How Memory is Allocated?



 So, From the Above fig. We can Conclude –

 Union Members that compose a union, all share the same storage area within the computer’s memory Each member within a structure is assigned its own unique storage area. Thus, unions are used to observe memory.

 Unions are useful for application involving multiple members, where values need not be assigned to all the members at any one time.

Accessing union members

 While accessing union, we can have access to single data member at a time. we can access single union member using following two Operators –

* 1. Using DOT Operator
  2. Using ARROW Operator

1. Accessing union members DOT operator

 In order to access the member of the union we are using the dot operator. DOT operator is used inside printf and scanf statement to get/set value from/of union member location.

 Syntax : variable\_name.member

 Consider the below union, when we declare a variable of union type then we will be accessing union members using dot operator.

 unionemp {

int id; charname[20];

}e1;

 id can be Accessed by – union\_variable.member

|  |  |
| --- | --- |
| Syntax | Explanation |
| e1.id | Access id field of union |
| e1.name | Access name field of union |

1. Accessing union members Arrow operator

 Instead of maintain the union variable suppose we store union at particular address then we can access the members of the union using pointer to the union and arrow operator.

id can be Accessed by – union\_variable->member



unionemp

{

int id; charname[20];

}\*e1;

|  |  |
| --- | --- |
| Syntax | Explanation |
| e1->id | Access id field of union |
| e1->name | Access name field of union |

|  |  |
| --- | --- |
| Program 1: Using dot operator #include <iostream> using namespace std;  union emp {  int id; charname[20];  }e1;  intmain(){  e1.id=10;  cout<<"\n ID:"<<e1.id; strcpy(e1.name,"Pritesh"); cout<<"\n Name:"<<e1.name; return 0;  } | Program 2: Accessing same memory #include <iostream> using  namespace std; union emp {  int id; charname[20];  }e1;  intmain(){  e1.id = 10; strcpy(e1.name,"Pritesh"); cout<<"\n ID : "<<e1.id; cout<<"\n Name : "<<e1.name; return 0;  } |
| Output :  ID : 10  Name :Pritesh | Output :  ID : 1953067600 Name  :Pritesh |

 As we already discussed in the union basics, we have seen how memory is shared by all union fields. In the above example –

 Total memory for union = max(sizeof(id), sizeof(name))

= sizeof(name)

= 20 bytes

 Firstly, we have utilized first two bytes out of 20 bytes for storing integer value. After execution of statement again same memory is overridden by character array so while printing the ID value, garbage value gets printed

1. Enumeration

 An enumeration is a user-defined data type that consists of integral constants. To define an enumeration, keyword enum is used.

 Syntax: enum flag {const1, const2, ..., constN};

 Here, name of the enumeration is flag. And, const1, const2. , constN are values of

type flag.

 By default, const1 is 0, const2 is 1 and so on. You can change default values of enum elements during declaration (if necessary).

* // Changing default values of enum enum suit {

club = 0,

diamonds = 10,

};

* Enumerated Type Declaration

 When you create an enumerated type, only blueprint for the variable is created.

Here's how you can create variables of enum type. enum boolean {false, true}; enum boolean check;

 Here, a variable check of type enum boolean is created.

 Here is another way to declare same check variable using different syntax. enum boolean

{

false, true

} check;

|  |
| --- |
| Live Example for Enumeration |
| #include<iostream> using namespace std;  enum week {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}; int main()  {  enum week today; today =Wednesday;  cout<<"Day :"<<today+1; return0;  } |
| Program output:  Day 4 |

1. The typedef Concept:

 The C programming language provides a keyword called typedef, by using this keyword you can create a user defined name for existing data type.

 Generally, typedef are used to create an alias name (nickname).  Declaration of typedef

Syntax: typedef datatype alias\_name; Example: typedef int Intdata;

Live Example for typedef concept

|  |
| --- |
| #include<iostream> using namespace std; typedef int Intdata; // Intdata is alias name of int int main()  {  int a=10; Intdata b=20;  typedef int dataIntegerdata; // Integerdata is again alias name of Intdata Integerdata s; s=a+b;  cout<<"\n Sum:= "<<s;  } |
| Program output: Sum: 20 |

# 7. Algorithm for the problem Statement:

1. Begin
2. Define structure

typedef struct BankAccount { short atype;

long acno,pin;

string cname; double balance; bool ifacility;

};

1. Define array of structure variable of type BankAccount. BankAccount ca[25];
2. Define insert() function/subroutine for reading the account details for n customers
3. Define customerType() function/subroutine for Identifying the golden, silver and general customers. a. For i=0 to n-1:
   1. If ca[i]. balance>=1000000: print “Golden Customer”
   2. Else If ca[i]. balance>=500000 && ca[i]. balance<1000000: print” Silver Customer”
   3. Else If ca[i]. balance<500000: print “General Customer”
4. Define InternetFacility() function/subroutine to display the list of customers availing the Internet banking facility. a. For i=0 to n-1:
   1. If ca[i]. ifacility ==1:

print “Customer Details like account\_no name”

1. Define CustomerAType() function/subroutine for displaying the customer list as per their account type. a. For i=0 to n-1:
   1. If ca[i]. atype=1:

print “Saving Account Customer”

* 1. Else If ca[i]. atype=2: print “Recurring Account Customer”
  2. Else If ca[i]. atype=3: print “Deposit Account Customer”

1. Define AreaPinSearch() function/subroutine for displaying the customers belonging to a particular geographical location
2. Read pincode from user for Searching purpose.
3. For i=0 to n-1:

i. If ca[i]. pin=pincode: print “Customer Details like account\_no, name” 9. Then call each subroutines one by one in sequence starting from inert().

1. Stop.
2. Flowchart

Note: student have to draw the flowchart for this assignment and attach to the assignment.

1. Implementation:

|  |
| --- |
| **C++ Program to perform Customer Bank account** |
| **#include<iostream> #include<iomanip> #include<string> using namespace std; int n;**  **typedef struct BankAccount { short atype;**  **long acno,pin; string cname;**  **long double balance; bool ifacility;**  **};**  **BankAccount ca[25];**  **void insert()//a. Read account details for n customers**  **{**  **ab: cout<<"\n Enter how many Customers:"; cin>>n;**  **if(n>25 || n<0)**  **{**  **cout<<"\n Enter number of Customers inbetween 1 to 25**  **only.";**  **goto ab;**  **}**  **cout<<"\n\t Enter "<< n<< " number of Customer details:"<<endl;**  **for (int i = 0; i < n; i++) { cout<<"\n\t Account Number:";**  **cin>>ca[i].acno;** |

**cout<<"\n\t Name:"; cin>>ca[i].cname; cout<<"\n\t Balance:"; cin>>ca[i].balance;**

**cout<<"\n\t Is customer using Internet Banking facility(1: Yes,0: NO):";**

**cin>>ca[i].ifacility; cout<<"\n\t Pincode:"; cin>>ca[i].pin;**

**cout<<"\n\t Account Type(1:Saving, 2: Recurring, 3: Deposit):"; cin>>ca[i].atype;**

**}**

**}**

**void InternetFacility()//Display the list of customers availing the Internet banking facility**

**{**

**cout<<"\n "<<endl;**

**cout<<"\n\t -------- Internet Banking Facility Customers "<<endl;**

**cout<<setw(10)<<"A Number"<<setw(20)<<"Customer Name"<<setw(10)<<" Balance"<<setw(10)<<"Pincode"<<endl;**

**cout<<"\n "<<endl;**

**for (int i = 0; i < n; i++) {**

**//Is customer using Internet Banking Facility if(ca[i].ifacility)**

**cout<<setw(10)<<ca[i].acno<<setw(20)<<ca[i].cname**

**<<setw(10)<<ca[i].balance<<setw(10)<<ca[i].pin<<”\n”;**

**}**

**}**

**void CustomerCategory() //Identify the golden, silver and general customers.**

**{**

**cout<<"\n\t All Custoer Details based on Balance:"<<endl;**

**cout<<"\n-------------------------------------------------------------------------------**

**"<<endl;**

**cout<<"\n\t -------- Golden Customers "<<endl;**

**cout<<setw(10)<<"A Number"<<setw(20)<<"CustomerName"<<setw(10)**

**<<" Balance"<<setw(10)<<"Pincode"<<endl;**

**cout<<"\n "<<endl;**

**for (int i = 0; i < n; i++) {**

**//Golden customers: Balance> 10,00000 if(ca[i].balance>=1000000)**

**cout<<setw(10)<<ca[i].acno<<setw(20)<<ca[i].cname**

**<<setw(10)<<ca[i].balance <<setw(10)<<ca[i].pin<<endl;**

**}**

**cout<<"\n -------------------------------------------------------------------------------**

**"<<endl;**

**cout<<"\n\t -------- Silver Customers "<<endl;**

**cout<<setw(10)<<"A Number"<<setw(20)<<"Customer Name"**

**<<setw(10)<<" Balance"<<setw(10)<<"Pincode"<<endl;**

**cout<<"\n -------------------------------------------------------------------------------**

**"<<endl;**

**for (int i = 0; i < n; i++){**

**//Golden customers: Balance> 500000 and Balance< 1000000 if(ca[i].balance<1000000 && ca[i].balance>500000)**

**cout<<setw(10)<<ca[i].acno<<setw(20)<<ca[i].cname<<setw(10)**

**<<ca[i].balance<<setw(10)<<ca[i].pin<<endl; }**

**cout<<"\n -------------------------------------------------------------------------------**

**"<<endl;**

**cout<<"\n\t -------- General Customers "<<endl;**

**cout<<setw(10)<<"A Number"<<setw(20)<<"Customer Name"**

**<<setw(10)<<" Balance"<<setw(10)<<"Pincode"<<endl;**

**cout<<"\n "<<endl;**

**for (int i = 0; i < n; i++) {**

**//Golden customers: Balance< 500000 if(ca[i].balance<=500000)**

**cout<<setw(10)<<ca[i].acno<<setw(20)<<ca[i].cname<<setw(10)**

**<<ca[i].balance<<setw(10)<<ca[i].pin<<endl;**

**}**

**}**

**void CustomerAType()//Display the customer list as per their account type.**

**{**

**cout<<"\n\t All Custoer Details based on Account Type"<<endl;**

**cout<<"\n "<<endl;**

**cout<<"\n\t -------- Saving Account Type Customers "<<endl;**

**cout<<setw(10)<<"A Number"<<setw(20)<<"Customer Name"**

**<<setw(10)<<" Balance"<<setw(10)<<"Pincode"<<endl;**

**cout<<"\n "<<endl;**

**for (int i = 0; i < n; i++) { if(ca[i].atype==1) //Saving Account Type**

**cout<<setw(10)<<ca[i].acno<<setw(20)<<ca[i].cname<<setw(10)<<ca[i].balance**

**<<setw(10)<<ca[i].pin<<endl;**

**}**

**cout<<"\n "<<endl;**

**cout<<"\n\t -------- Recurring Account Type Customers----------**

**"<<endl;**

**cout<<setw(10)<<"A Number"<<setw(20)<<"Customer Name"**

**<<setw(10)<<" Balance"<<setw(10)<<"Pincode"<<endl;**

**cout<<"\n "<<endl;**

**for (int i = 0; i < n; i++) {**

**if(ca[i].atype==2) //Recurring Account Type**

**cout<<setw(10)<<ca[i].acno<<setw(20)<<ca[i].cname<<setw(10)<<ca[i].balance**

**<<setw(10)<<ca[i].pin<<endl;**

**}**

**cout<<"\n -------------------------------------------------------------------------------**

**"<<endl;**

**cout<<"\n\t -------- Deposit Account Type Customers "<<endl;**

**cout<<setw(10)<<"A Number"<<setw(20)<<"Customer Name"<<setw(10)**

**<<" Balance"<<setw(10)<<"Pincode"<<endl;**

**cout<<"\n "<<endl;**

**for (int i = 0; i < n; i++) {**

**if(ca[i].atype==3) //Deposit Account Type**

**cout<<setw(10)<<ca[i].acno<<setw(20)<<ca[i].cname<<setw(10)<<ca[i].balance**

**<<setw(10)<<ca[i].pin<<endl;**

**}**

**}**

**void AreaPinSearch()//Display customers belonging to a particular geographical location**

**{**

**long tempin;**

**cout<<"\n\t Enter the Pincode for Search:";**

# Output:

Enter how many Customers:3

Enter 3 number of Customer details:

Account Number:10111 Name:Kuldeep\_Hule Balance:1200000

Is customer using Internet Banking facility(1: Yes,0: NO):1 Pincode:422011

Account Type(1:Saving, 2: Recurring, 3: Deposit):2 Account Number:10112

Name:Rohan\_Shinde Balance:650000

Is customer using Internet Banking facility(1: Yes,0: NO):1 Pincode:422012

Account Type(1:Saving, 2: Recurring, 3: Deposit):1 Account Number:10113

Name:Jay\_Sharma Balance:300000

Is customer using Internet Banking facility(1: Yes,0: NO):0 Pincode:422012

Account Type(1:Saving, 2: Recurring, 3: Deposit):3 All Custoer Details based on Balance:

-------- Golden Customers----------

A Number Customer Name Balance Pincode

10111 Kuldeep\_Hule 1200000 422011

-------- Silver Customers----------

A Number Customer Name Balance Pincode

10112 Rohan\_Shinde 650000 422012

-------- General Customers----------

A Number Customer Name Balance Pincode

10113 Jay\_Sharma 300000 422012

--- -------- Internet Banking Facility Customers----------

A Number Customer Name Balance Pincode

--- 10111 Kuldeep\_Hule 1200000 422011

10112 Rohan\_Shinde 650000 422012

All Custoer Details based on Account Type

--- -------- Saving Account Type Customers A

Number Customer Name Balance Pincode

--- 10112 Rohan\_Shinde 650000 422012

--- -------- Recurring Account Type Customers----------

A Number Customer Name Balance Pincode

|  |
| --- |
| ---  10111 Kuldeep\_Hule 1200000 422011 |
| --- -------- Deposit Account Type Customers A  Number Customer Name Balance Pincode |
| --- 10113 Jay\_Sharma 300000 422012  Enter the Pincode for Search:422012 |
| --- -------- Customers Details ----------  A Number Customer Name Balance Pincode |
| --- 10112 Rohan\_Shinde 650000 422012 10113  Jay\_Sharma 300000 422012 |

# Conclusion:

From above experiment I have conclude that:

**We are able to define data types using the concept of Structure & array of structure.**

1. **Review Questions & Exercises:**
   1. **Fill in the Blanks**
      1. Before a structure variable can be created, the structure must be \_ **Declared** .
      2. In the definition of a structure variable, the **“struct” keyword** is placed before the variable name, just like the data type of a regular variable is placed before its name.
      3. The variables declared inside a structure declaration are called **Structure Members/ Fields .**

# Answer the following:

* + 1. **Look at the following code.**

struct Town {

string townName; string countyName; double population; double elevation;

};

Town t = { "Canton", "Haywood", 9478 };

* + - 1. What value is stored in t.townName ?

**"Canton"**

* + - 1. What value is stored in t.countyName ?

**"Haywood"**

* + - 1. What value is stored in t.population ?

**9478**

* + - 1. What value is stored in t.elevation ?

**0**

* + 1. **Look at the following declaration.**

enum Person { BILL, JOHN, CLAIRE, BOB };

Person p;

Indicate whether each of the following statements or expressions is valid or invalid.

* + - 1. p = BOB; **// Valid**
      2. p ++ ; **// Valid**
      3. BILL > BOB **// Invalid**
      4. p = 0; **// Valid**
      5. int x = BILL; **// Invalid**
      6. p = static\_cast<Person>(3); **// Invalid**
      7. cout << CLAIRE << endl; **// Valid**

# Algorithm Workbench

The structure Car is declared as follows: struct Car {

string carMake; string carModel; int yearModel; double cost;

};

# Write a definition statement that defines a Car structure variable initialized with the following data:

Make: Ford Model: Mustang Year Model: 1968 Cost: $20,000

**Answer:**

**Car Car1 = {“Ford”, “Mustang”, 1968, 20000};**

* + 1. **Define an array of 35 of the Car structure variables. Initialize the first three elements with the following data:**

|  |  |  |  |
| --- | --- | --- | --- |
| Make | Model | Year | Cost |
| Ford | Taurus | 1997 | 11,00,000 |
| Honda | Accord | 1992 | 5,00,000 |
| Lamborghini | Countach | 1997 | 20,00,000 |
| **Answer** |  |  |  |

**Car Cars[35];**

**Cars[0] = {“Ford”, “Taurus”, 1997, 1100000};**

**Cars[1] = {“Honda”, “Accord”, 1992, 500000};**

**Cars[2] = {“Lamborghini”, “Countach”, 1997, 2000000};**

* + 1. **Write a loop that will step through the array you defined in Question ii, displaying the contents of each element.**

**Answer**

**for (int i = 0; i < 3; i++) {**

**cout << “Make : ” << Cars[i].carMake << “\t”**

**<< “Model : ” << Cars[i].carModel<< “\t”**

**<< “Year : ” << Cars[i].yearModel << “\t”**

**<< “Cost : ” << Cars[i].cost << endl;**

**}**

* 1. **Programming Assignments:**
     1. Write a program in C++ using structure for maintaining extra-curricular activities of students (roll, name, year, activity name, and prize). The prize can be either cash prize or memento but not both. Cash prize is to be recorded as integer and memento is to be recorded as character string. Use union within structure for prize. Read extra- curricular activity record for n students and Display extra-curricular activities.

**Solution**

**#include<iostream> #include<iomanip>**

**using namespace std; int n;**

**union prize{**

**int cash;**

**char momento[50];**

**};**

**struct cca\_activity {**

**string name, activity\_name; int year, roll\_no, prize\_choice; union prize prize\_to\_give;**

**}cca[100];**

**void input\_data()**

**{**

**cout<<"\n Enter the Activity details for : \n"; for( int i = 0; i < n ; i++)**

**{**

**cout<<"\n "<<i+1<<" student "; cout<<"\n Roll no. : "; cin>>cca[i].roll\_no;**

**cout<<" Name : "; cin>>cca[i].name; cout<<" Year : "; cin>>cca[i].year;**

**cout<<" Activity Name : "; cin>>cca[i].activity\_name;**

**cout<<" Activity Prize : 0) Cash or 1) Momento : "; cin>>cca[i].prize\_choice;**

**cout<<" Enter Prize : \t"; if (cca[i].prize\_choice == 1)**

**cin>>cca[i].prize\_to\_give.cash;**

**else**

**cout<<endl;**

**}**

**}**

**cin>>cca[i].prize\_to\_give.momento;**

**int main()**

**{**

**cout<<"\n Enter the number of participants : "; cin>>n;**

**input\_data();**

**cout<<"\n \n";**

**cout<<setw(10)<<"RollNo"<<setw(20)<<"Name"<<setw(20)**

**<<"Year"<<setw(20)<<"Activity Name"<<setw(20)<<"Prize"<<endl; for( int i =0 ; i < n; i++)**

**{**

**cout<<setw(10)<<cca[i].roll\_no<<setw(20)<<cca[i].name<<setw(20)**

**<<cca[i].year<<setw(20)<<cca[i].activity\_name<<setw(20); if (cca[i].prize\_choice == 0)**

**cout<<cca[i].prize\_to\_give.cash<<endl;**

**else**

**}**

**cout<<cca[i].prize\_to\_give.momento<<endl;**

**return 0;**

**}**

**OUTPUT:**

* + 1. Write a program in C++ to define a structure that holds Information of items like ItemNumber, Item Names, Item Category (Electronics, Food, Cosmetics etc), price, manufacturing date & Available stock/qty. Display the available items.

**Solution**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct store**

**{**

**int ItemNumber;**

**string ItemNames;**

**int ItemCateogory;**

**double price;**

**string manufacturing\_date;**

**int stockQuantity;**

**};**

**store s[25];**

**int n;**

**void display\_data(){**

**cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"<<endl;**

**cout<<"ITEMS IN THE SHOP"<<endl;**

**for(int j=0;j<n;j++){**

**cout<<"ITEM NUMBER:- "<<s[j].ItemNumber<<endl;**

**cout<<"ITEM NAME:- "<<s[j].ItemNames<<endl;**

**cout<<"ITEM MANUFACTURING DATE:- "<<s[j].manufacturing\_date<<endl;**

**cout<<"ITEM PRICE:- "<<s[j].price<<endl;**

**if(s[j].ItemCateogory==1){**

**cout<<"ITEM CATEOGORY IS:- ELECTRONICS"<<endl;**

**}**

**if(s[j].ItemCateogory==2){**

**cout<<"ITEM CATEOGORY IS:- FOOD"<<endl;**

**}**

**if(s[j].ItemCateogory==3){**

**cout<<"ITEM CATEOGORY IS:- COSMETICS"<<endl;**

**}**

**if(s[j].stockQuantity==0){**

**cout<<"ITEM NOT AVAILABLE."<<endl;**

**}**

**if(s[j].stockQuantity!=0){**

**cout<<"ITEM QUANTITY IS:- "<<s[j].stockQuantity<<endl;**

**}**

**cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;**

**}**

**}**

**void input\_data()**

**{**

**cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;**

**cout<<endl;**

**cout<<"Please Enter the number of entries:- ";**

**cin>>n;**

**cout<<"\t\tPlease Provide the information for the shop."<<endl;**

**for(int i=0;i<n;i++){**

**cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;**

**cout<<"Entry Number:- "<<i+1<<endl;**

**cout<<"Enter the Item number:- ";**

**cin>>s[i].ItemNumber;**

**cout<<"Enter the Name of the Item:- ";**

**cin>>s[i].ItemNames;**

**cout<<"Please Enter the type of item(1.Electronics 2.Food 3,Cosmetics):- ";**

**cin>>s[i].ItemCateogory;**

**cout<<"Enter the price of the item:- ";**

**cin>>s[i].price;**

**cout<<"Enter the manufacturing date in the format dd/mm/yy:- ";**

**cin>>s[i].manufacturing\_date;**

**cout<<"Enter the quantity of the item:- ";**

**cin>>s[i].stockQuantity;**

**}**

**}**

**int main()**

**{**

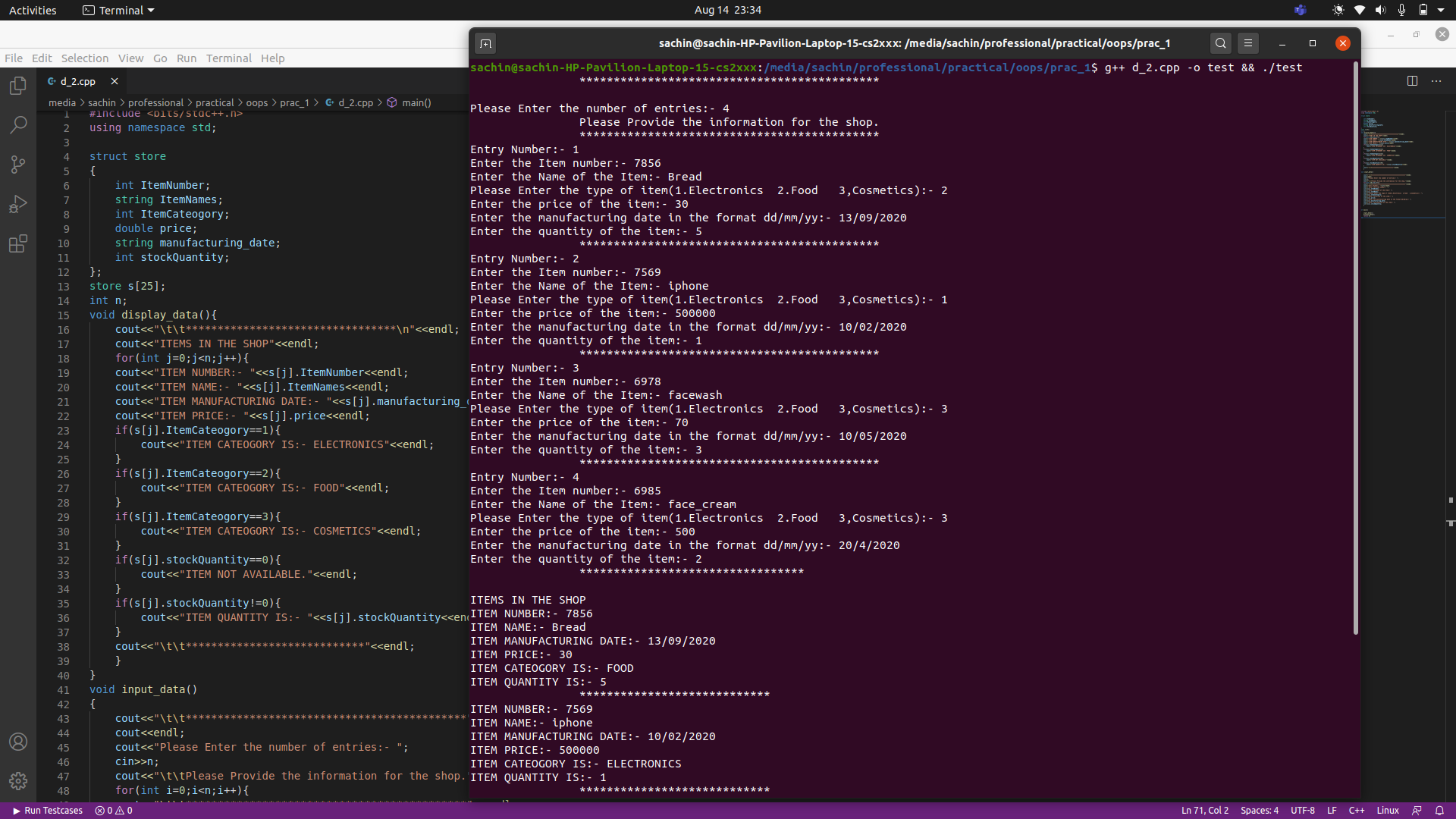
**input\_data();**

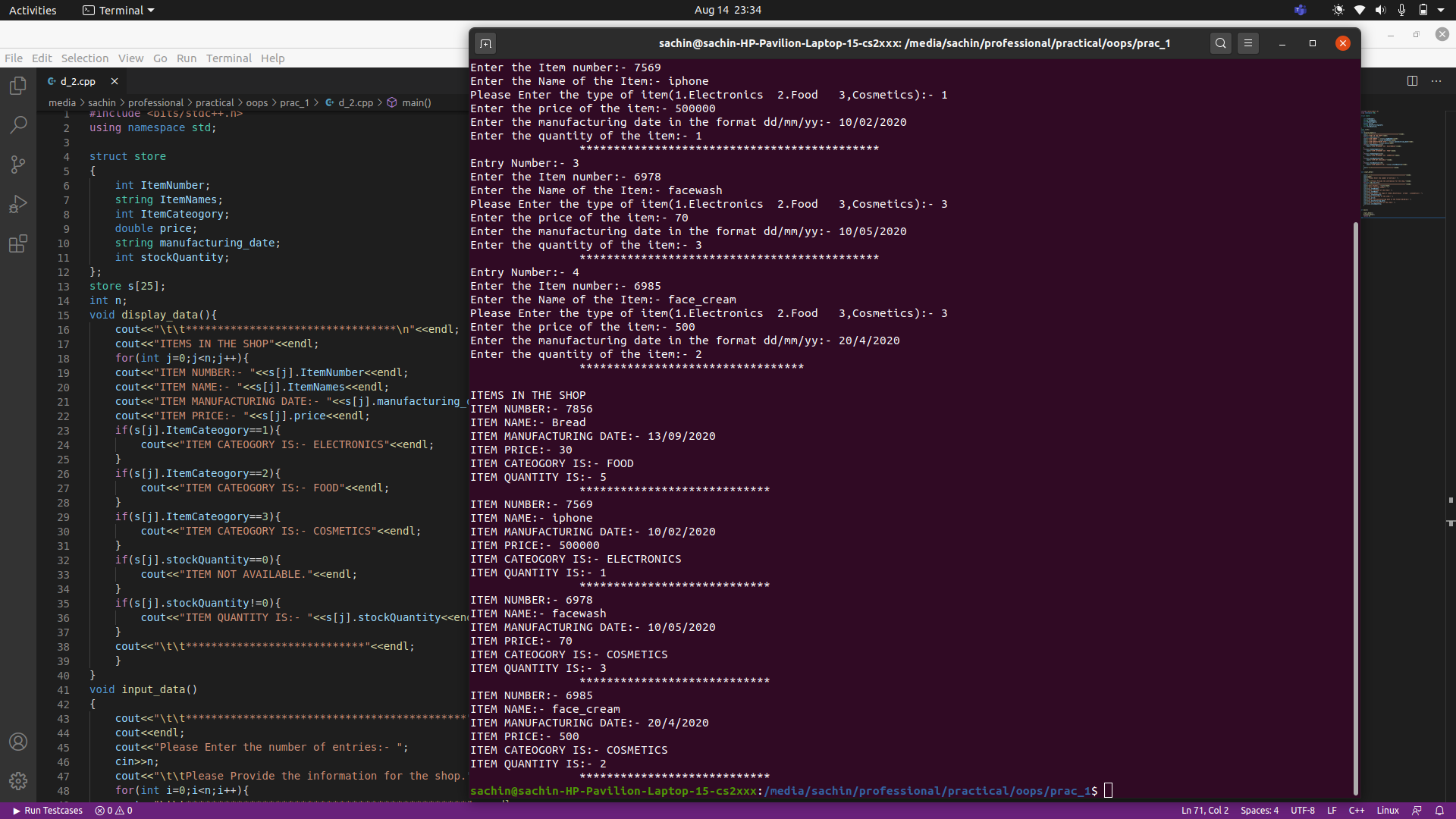
**display\_data();**

**return 0;**

**}**

**OUTPUT**





3.Write a program that simulates a soft drink machine. The program should use a structure that stores the following data:Drink Name, Drink Cost, Number of Drinks in Machine

The program should create an array of five structures. The elements should be initialized with the following data:

|  |  |  |
| --- | --- | --- |
| Drink Name | Cost | Number in Machine |
| Cola | 25 | 20 |
| Root Beer | 25 | 20 |
| Lemon-Lime | 25 | 20 |
| Grape Soda | 50 | 20 |
| Cream Soda | 50 | 20 |

Each time the program runs, it should enter a loop that performs the following steps:

1. A list of drinks is displayed on the screen.
2. The user should be allowed to either quit the program or pick a drink.
3. If the user selects a drink, he or she will next enter the amount of money that is to be inserted into the drink machine. The program should display the amount of change that would be returned and subtract one from the number of that drink left in the machine.
4. If the user selects a drink that has sold out, a message should be displayed.
5. The loop then repeats.
6. When the user chooses to quit the program, it should display the total amount of money the machine earned.

Input Validation: When the user enters an amount of money, do not accept negative values or values greater than 150.

**SOLUTION**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct drink\_machine**

**{**

**string drink\_name;**

**int drink\_cost;**

**int number\_of\_drinks;**

**};**

**drink\_machine m[5];**

**int main()**

**{**

**char d='Y';**

**cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;**

**cout<<"\t\tDRINK MACHINE STIMULATOR"<<endl;**

**m[0].drink\_name="Cola";**

**m[0].drink\_cost=25;**

**m[0].number\_of\_drinks=20;**

**m[1].drink\_name="Froot Bear";**

**m[1].drink\_cost=25;**

**m[1].number\_of\_drinks=20;**

**m[2].drink\_name="Lemon Lime";**

**m[2].drink\_cost=25;**

**m[2].number\_of\_drinks=20;**

**m[3].drink\_name="Grape Soda";**

**m[3].drink\_cost=50;**

**m[3].number\_of\_drinks=20;**

**m[4].drink\_name="Cream Soda";**

**m[4].drink\_cost=50;**

**m[4].number\_of\_drinks=20;**

**cout<<"Drinks Available:- "<<endl;**

**for(int i=0;i<5;i++){**

**cout<<i+1<<"."<<m[i].drink\_name<<" Rs-"<<m[i].drink\_cost<<endl;**

**}**

**int choice;**

**int amount;**

**int change;**

**double moneyEarned=0;**

**while(d!='N'){**

**cout<<"Which drink you want to buy:- ";**

**cin>>choice;**

**cout<<"You choose "<<m[choice-1].drink\_name<<endl;**

**if(m[choice-1].number\_of\_drinks==0){**

**cout<<"Drink not available.Choose another drink."<<endl;**

**continue;**

**}**

**cout<<"Enter the amount of money you insert in machine:- ";**

**cin>>amount;**

**if(amount<0 || amount>150){**

**cout<<"Amount not acceptable.Please Enter valid amount between (1,150)"<<endl;**

**continue;**

**}**

**change=amount-(m[choice-1].drink\_cost);**

**cout<<"Change you get from your paid amount is:- "<<change<<endl;**

**m[choice-1].number\_of\_drinks-=1;**

**moneyEarned+=(m[choice-1].drink\_cost);**

**cout<<"\nDo you want to continue(Y/N):- ";**

**cin>>d;**

**if(d=='N' || d=='n'){**

**cout<<"\n\n\t\tThank You"<<endl;**

**break;**

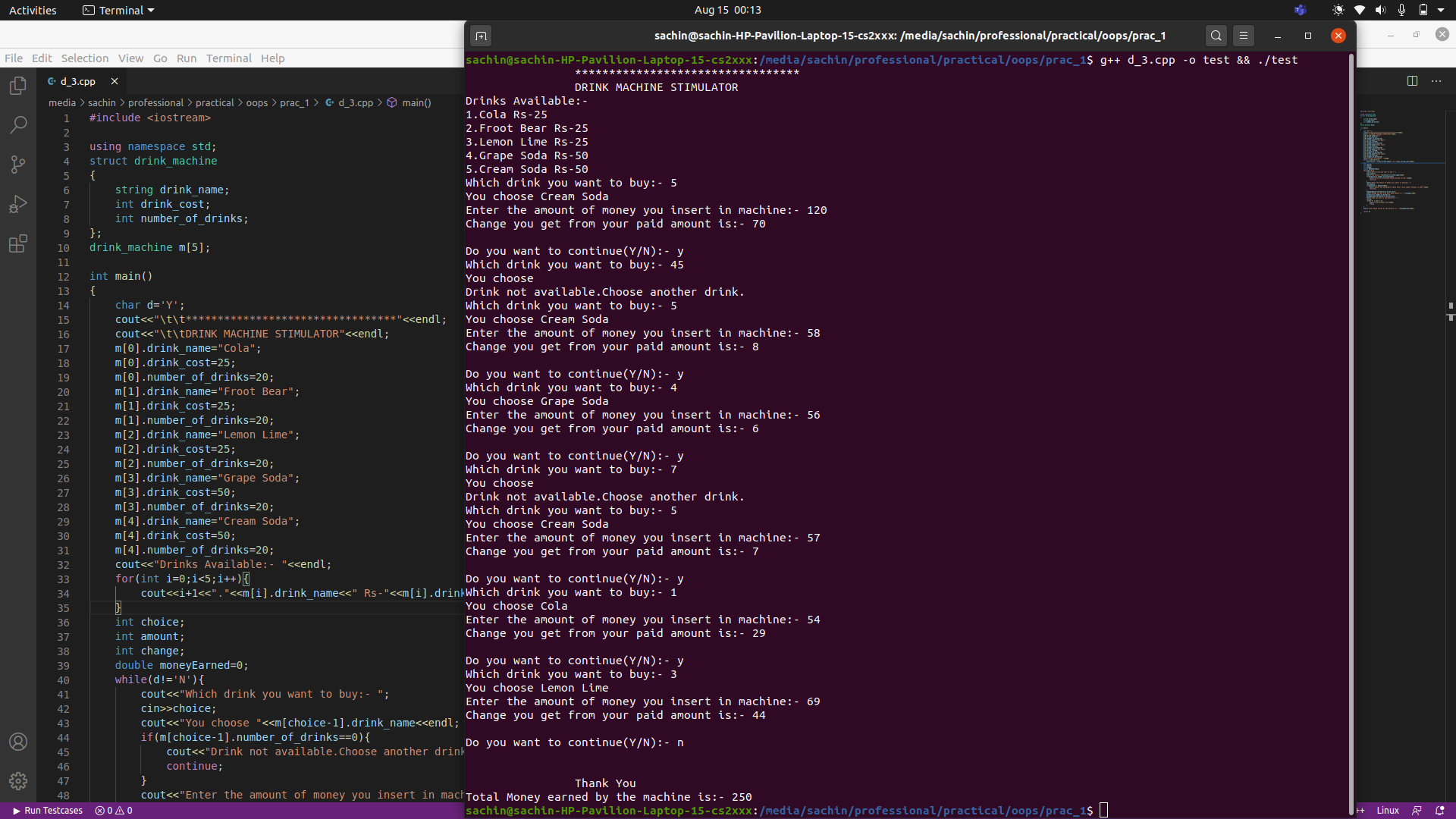
**}**

**}**

**cout<<"Total Money earned by the machine is:- "<<moneyEarned<<endl;**

**return 0;**

**}**

****

1. **References:**
2. E Balagurusamy Object-Oriented Programming with C++.7th edition. McGraw-

Hill Publication, ISBN 10: 9352607996 ISBN 13: 9789352607990

1. Tony Gaddis- “STARTING OUT WITH C++ From Control Structures through Objects”, Pearson Education, ISBN 13: 978-0-13-376939-5