**HAFIZ AHMAD  
19l1316**

**EE-2A**

**LAB REPORT AND POST LAB:**

**Experiment No.09**

**Value and Reference parameters in Functions**

**OBJECTIVE**:

 To get an understanding of reference parameters

 To learn the difference between value and reference parameters

***Functions with Value Parameters***

Function Declaration: A value parameter is used to pass information ***into*** a function to be processed. Remember the general form for a function declaration:

type function-name (formal parameter type list);

A void function with value parameters are declared by enclosing the list of types for the parameter list in the parentheses.

**Example:** A function that prints out a user specified number of horizontal lines is declared as:

// Purpose: Print out a number of lines

// Precondition: numOfLines has a value assigned.

// Postcondition: the number of lines are printed.

void PrintLines(int);

Function Activation: To activate a void function with value parameters, we specify the name of the function and provide the actual arguments enclosed in parentheses. The order and types of the list of arguments should correspond exactly to those of the formal parameters declared in the function prototype. The arguments can be constants, expressions, variables, or even function calls themselves. If an argument is a variable, at the time of function activation, the variable must have a value.

**Example:**

PrintLines(8);

or

numOfLines = 5;

PrintLines(numOfLines);

***Void Functions with Reference Parameters***

Function Declaration: The second type of parameter in C++ is called a **reference parameter**. These parameters are used to send back a value (***output)***, or both to send in and out values (***input and output)*** from functions. Reference parameters have the ampersand (**&**) following their type identifier in the function prototype and function heading.

**Example:** A function that reads and returns (**output**) the length and width of a rectangle entered by the user, is declared as:

// Purpose: Read and return the length and width of a rectangle

// Postcondition: the values of the two arguements are read

void GetData(int& length, int& width);

**Exercise – 1 (10 points)**

We need to design software for an ATM machine. An ATM card is used for money withdrawal. The systems needs following information of a card: Introduction to Computing Lab Manual

Page | 46

 Card Number

 Initial balance

 Type of card. (gold and silver)

**InputCardDetails( .. ):** This function should prompt the user to enter relevant details of ATM Card. Decide the prototype yourself. Remember that variables must be declared in main only. **The output from function should be displayed on console through main**.



Write the main and test your program.

**Exercise – 2 (10 points)**

In the previous exercise add the with draw money functionality.

**WithdrawMoney(..):** This function should take all the details of cards as input and ask the user the amount he/she wants to withdraw. Following rules should be observed:

 For Silver card: Money to be withdrawn cannot be more than 10000.

 For Gold card: Money to be withdrawn cannot be more than 25000.

*Your program should keep displaying the menu after each function call unless user wants to exit.*

Algorithm of main program is given below. Convert it into C++ to test your functions accordingly

 Declare all required variables in main

 Get input of a card from user

 Display the menu

 Withdraw some amount

**Exercise – 3 (10 points)**

In the previous exercise add the functionality of balance enquiry or transfer money to other bank. For this add the following functions:

**BalanceInquiry(..):** This function generates a message that displays card #, and current balance.

**TransferMoney(..):** This function should ask the user the account number to which amount has to be transferred. It should keep a check on the balance before transferring any amount.

*Your program should keep displaying the menu after each function call unless user wants to exit.*

Algorithm of main program is given below. Convert it into C++ to test your functions accordingly

 Declare all required variables in main

 Get input of a card from user

 Display the menu

 Withdraw some amount

 Generate an inquiry about the balance

 Transfer some amount

 Generate an inquiry about the balance

**Code:**

#include <iostream>

#include <string>

using namespace std;

void input(int a,int b,string c)

{

cout<<"enter your card number"<<endl;

cin>>a;

cout<<"enter your initial balance"<<endl;

cin>>b;

cout<<"enter your card type (silver) or (gold)"<<endl;

cin>>c;

}

void output(int number,int balance,string type)

{

cout<<"card number"<<number<<endl;

cout<<"initil balance"<<balance<<endl;

cout<<"card type"<<endl;

}

void withdrawalmoney(int aa,int bb,string cc)

{

int limit,withdraw;

cout<<"enter the ammount to withdraw"<<endl;

cin>>withdraw;

if(cc=="silver")

{

limit=10000;

}

if(cc=="gold")

{

limit=25000;

}

if(withdraw>bb)

{

cout<<"not much money"<<endl;

}

else if(withdraw>limit)

{

cout<<"ammount is greater than limit"<<endl;

}

else

bb=bb-withdraw;

}

void checkbalance(int no,int balnce)

{

cout<<"your card number="<<endl;

cout<<"your balance is $"<<balnce<<endl;

}

void transfer(int ammount)

{

int aaa,moneytransfer;

cout<<"enter the card number"<<endl;

cin>>aaa;

cout<<"enter the money transfer"<<endl;

cin>>moneytransfer;

if (moneytransfer>ammount)

{

cout<<"not much money n account"<<endl;

}

else

ammount=ammount-moneytransfer;

}

void main()

{

int cardno,a1;

int inibal;

string crdtype;

cout<<"enter the number"<<endl;

cout<<"1:input card details"<<endl;

cout<<"2:output card deatil"<<endl;

cout<<"3:wihdraw money"<<endl;

cout<<"4:balance inquiry"<<endl;

cout<<"5:tranfer money"<<endl;

cin>>a1;

for(int i=1;i>=0;i++)

{ if(a1==1)

{

input(cardno,inibal,crdtype);

}

else if(a1==2)

{

output(cardno,inibal,crdtype);

}

else if(a1==3)

{

withdrawalmoney(cardno,inibal,crdtype);

}

else if(a1==4)

{

checkbalance(cardno,inibal);

}

else if(a1==5)

{

transfer(inibal);

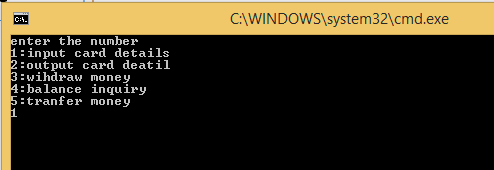
}

else

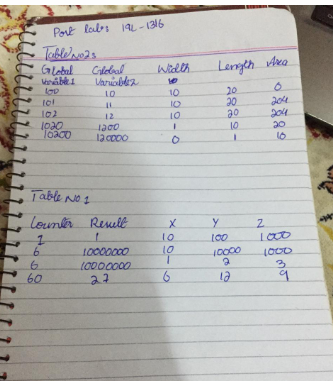
cout<<"limt exceed"<<endl;

break;

}

}

**POST LAB:**



**Conclusion:**

**IN this lab we learn to understand value reference parameter and by use of functions we make do our work easily.**