**Experiment 1**

**Aim:** To learn and implement the concept of complex nesting loops

# **Problem Definition:** To enter five strings and then display them as top,bottom,right and left justified , as you want.

# Input given: 5 String

# Output received: Appropriately justified strings.

**Algorithm Description:**

MAIN PROGRAM:

**Step 1:** START

**Step 2:** Create an array of class Names of length 5.Input name strings into it.

**Step 3:** Menu displayed with justification options. Required option number is entered.

**Step 4:**  Switch case executes the appropriate function.

**Step 5:** END

topJust() FUNCTION:

**Step 1:** START

**Step 2:** Create Initiate i=0 and start a loop while i<10.Initiate k=0.

**Step 3:** If i<length of the string, kth char of string is stored in 2D array ,else blank space is stored. Increment k by 1. Increment i by 1.

**Step 4:**  END

bottomJust() FUNCTION:

**Step 1:** START

**Step 2:** Create Initiate i=0 and start a loop while i<10.Initiate k=0.

**Step 3:** If 10- i>length of the string, blank space is stored, else kth char is stored in 2D array. Increment k by 1. Increment i by 1.

**Step 4:**  END

leftJust() FUNCTION:

**Step 1:** START

**Step 2:** Create Initiate i=0 and start a loop while i<10.Initiate k=0.

**Step 3:** If i<length of the string, kth char is store, else blank space is stored in 2D array. Increment k by 1. Increment i by 1.

**Step 4:**  END

rightJust() FUNCTION:

**Step 1:** START

**Step 2:** Create Initiate i=0 and start a loop while i<10.Initiate k=0.

**Step 3:** If 10-i>length of the string, blank space is stored, else kth char is store in 2D array. Increment k by 1. Increment i by 1.

**Step 4:**  END

**Code:**

#include<iostream>

using namespace std;

class Names{

public:

char n[10];

void getName();

};

void Names:: getName()

{

cin>>n;

}

int length(char b[])

{ int i=0;

while(b[i]!='\0')

{

int temp=0;

i++;

}

return i;

}

void bottomJust(char (\*A)[5] ,int j , char n[])

{

int i=0;

int k=0;

while(i<10)

{

if(10-i>length(n))

{

A[i][j]=' ';

i++;

}

else{

A[i][j]=n[k];

i++;k++;

}

}

}

void topJust( char (\*A)[5], int j, char n[])

{

int i=0;

int k=0;

while(i<10)

{

if(i<length(n))

{

A[i][j]=n[k];

i++;k++;

}

else{

A[i][j]=' ';

i++;

}

}

}

void rightJust(char (\*B)[10], int j, char n[])

{

int i=0,k=0;

while(i<10)

{

if(10-i>length(n))

{

B[j][i]=' ';

i++;

}

else{

B[j][i]=n[k];

i++;k++;

}

}

}

void leftJust(char (\*B)[10], int j, char n[])

{

int i=0,k=0;

while(i<10)

{

if(i<length(n))

{ B[j][i]=n[k];

i++;k++;

}

else{

B[j][i]=' ';

i++;

}

}

}

void printA(char A[10][5])

{ int i,j;

for(i=0;i<10;i++)

{

for(j=0;j<5;j++)

{

cout<<A[i][j];

}

cout<<endl;

}

}

void printB(char A[5][10])

{ int i,j;

for(i=0;i<5;i++)

{

for(j=0;j<10;j++)

{

cout<<A[i][j];

}

cout<<endl;

}

}

int main()

{

char A[10][5];

char B[5][10];

Names name[5];

cout<<"ENTER 5 NAMES";

name[0].getName();

name[1].getName();

name[2].getName();

name[3].getName();

name[4].getName();

int a;

cout<<"HOW DO YOU WANT TO PRINT THE NAMES?"<<endl;

cout<<"1.TOP JUSTIFIED"<<endl;

cout<<"2.BOTTOM JUSTIFIED"<<endl;

cout<<"3.RIGHT JUSTIFIED"<<endl;

cout<<"4.LEFT JUSTIFIED"<<endl;

cin>>a;

int j,i;

switch (a)

{

case 1:for(j=0;j<5;j++)

{topJust( A , j,name[j].n);}

printA(A);

break;

case 2:

for(j=0;j<5;j++)

{bottomJust( A , j,name[j].n);}

printA(A);

break;

case 3:for(j=0;j<5;j++)

rightJust( B , j ,name[j].n);

printB(B);

break;

case 4:for(j=0;j<5;j++)

leftJust( B , j ,name[j].n);

printB(B);

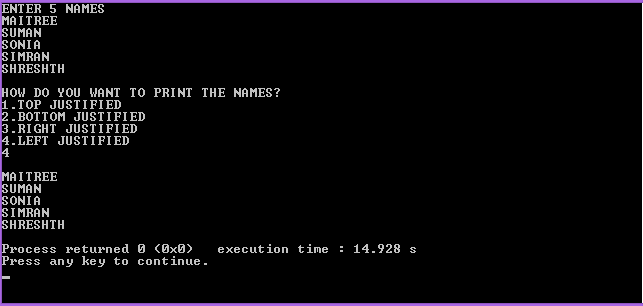
break;

}

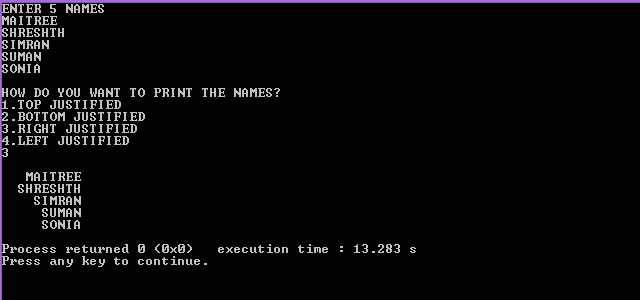
}

**Result:**

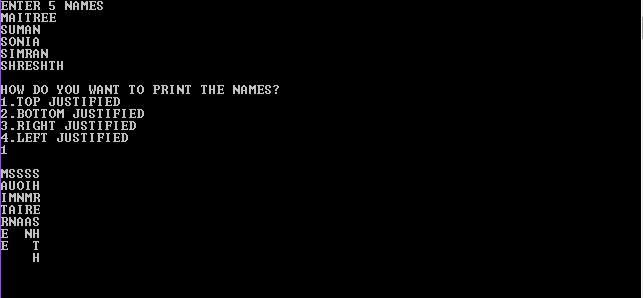
Left Justification-

****

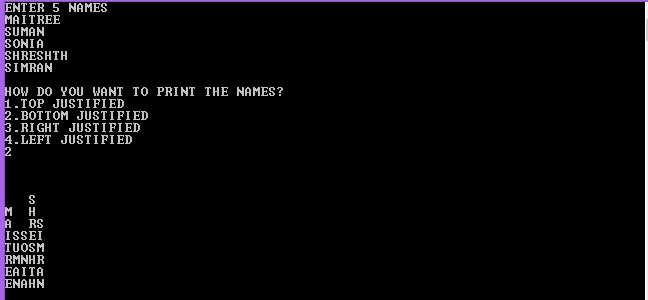
Right Justification-

****

Top Justification-

****

Bottom Justification-

****

**Discussion :**

Strategy Used- Enter five strings as class variables. A menu asks for what kind of justification is required. Then a switch case runs and the corresponding function to the choice made is executed.

**Conclusion:**

With the help of strings as character arrays ,loop and pointers , we have successfully displayed the strings correctly justified. No inbuilt library function was used here.

**Experiment 2**

# **Aim:** To implement the concept of reference variables and classes.

# **Problem Definition:** Write a program to find maximum of the two numbers using reference variable.

# Input given: Two numbers

# Output received: Maximum of the two

**Algorithm Description:**

**Step 1:** START

**Step 2:** Create a class with one data member value and a function getValue().

Create two objects. Create a function greatest to find the greatest of the two object values.

**Step 3:** Reference to the object values is passed as function arguments and compared.

**Step 4:** Larger of the two is printed.

**Step 5:** END

**Code:**

#include<iostream>

using namespace std;

class abc{

int value;

public:

int getValue();

void greatest(abc & , abc &);

};

int abc:: getValue()

{ cout<<"enter object value";

cin>> value;

return value;

}

void greatest(int & a, int & b)

{

a>b? cout<<a:cout<<b;

}

int main(){

abc obj1,obj2;

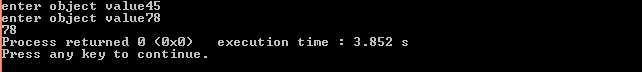
int n1 = obj1.getValue();

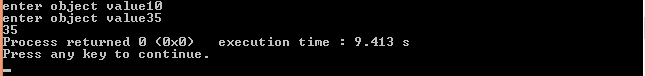
int n2 = obj2.getValue();

greatest(n1 , n2);

}

**Result:**





**Discussion**

A class is with two objects is created. The values of the objects are input by the user using getValue() function.The reference to these values are passed to the greatest() function and compared. The greatest of the two is printed.

**Conclusion:**

By using the reference variables , we have successfully compared the two values and printed the larger of the two. The changes made in the reference variables are made in the actual variables to which they point in the main program as well. Although in this case no changes are made.

**Experiment 3**

**Aim:** To implement the functions which have class objects as function arguments.

**Proble Definition:** Write a program to add two numbers using class by passing objects as function parameters..

# Input given: Two numbers as object values.

# Output received: Value of the third object.

**Algorithm Description:**

**Step 1:** START

**Step 2:** Create a class abc with one data member value and functions getValue() , Sum(abc , abc) and print(). Create three objects.

**Step 3:** Input the values of the first two objects using getValue() function.

**Step 4:** Call function Sum on object three.Pass first two objects as arguments to function Sum. Add their values and store in value of object three.

**Step 5:** Print value of object three.

**Step 6:** END.

**Code:**

#include<iostream>

using namespace std;

class abc{

int value;

public:

void getValue();

void printValue();

void sum(abc,abc);

};

void abc:: getValue()

{ cout<<"enter value";

cin>> value;

}

void abc:: printValue()

{

cout<<value;

}

void abc :: sum(abc a , abc b)

{

value= a.value + b.value;

}

int main(){

abc obj1,obj2,obj3;

obj1.getValue();

obj2.getValue();

obj3.sum(obj1,obj2);

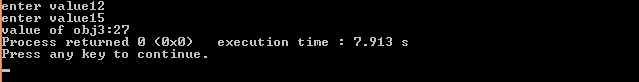
cout<< "value of obj3:";

obj3.printValue();

}

**Result:**

Screenshot (142).png



**Discussion :**

A class is with two objects is created. The values of the objects are input by the user using getValue() function. These objects are passed as function arguments to the function Sum() and the sum of their values is stored in the value of a third object. This value is then printed using print() function.

**Conclusion:**

By passing class objects as function arguments we have successfully added the values of two objects and stored it in the third object value.

**Experiment 4**

# **Aim:** To implement friend functions and classes.

# **Problem Definition:** Write a program to find greatest of two numbers which are objects of two different classes and use friend function to find the greatest of the two.

# Input given: Two numbers

# Output received: Greatest of the two

**Algorithm Description:**

**Step 1:** START

**Step 2:** Create tow classes with one data member value and a function getValue().

**Step 3:** create a friend function greatest() which takes two input parameter as objects of the two classes. Function returns the greatest of the two object values.

**Step 4:** Inthe main program create an object of each class and pass these objects to the greatest function. Greatest of the two is printed.

**Step 5:** END

**Code:**

#include<iostream>

using namespace std;

class A; //forward declaration

class B

{

int value;

public:

void getValue()

{

cout<<"\nENTER VALUE OF OBJECT B";

cin>>value;

}

friend int greatest(A,B);

};

class A

{

int value;

public:

void getValue()

{

cout<<"\nENTER VALUE OF OBJECT A";

cin>>value;

}

friend int greatest(A,B);

};

int greatest(A objA , B objB)

{

if(objA.value>objB.value)

{return objA.value;}

else

{return objB.value;}

}

int main(){

A obj1;

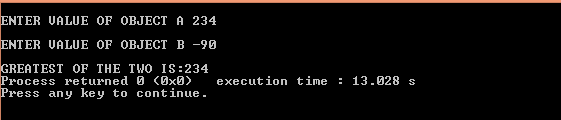
B obj2;

obj1.getValue();

obj2.getValue();

cout<<"\nGREATEST OF THE TWO IS:"<< greatest(obj1,obj2);}

**Result:**



**Discussion**

Two classes are created. Each of them has an external function greatest as friendly and hence it can access the private data members of the classes using dot operator. Objects of two classes are passed as function parameters to the friend function and the greatest of the two is returned and printed.

**Conclusion:**

By using the friend function , we have successfully compared the two values and printed the larger of the two.

**Experiment 5**

# **Aim:** To learn and implement array of objects

# **Problem Definition:** Write a program to accept the information of 10 employees and display their information using array of objects.

# Input given: Employee data.

# Output received: Employee data.

**Algorithm Description:**

**Step 1:** START

**Step 2:** Create a class employee with two void functions getData() and displayData().

**Step 3:** create Create in the main program an array of employees and get the value of each object from the user.

**Step 4:** Display these objects by calling display function.

**Step 5:** END

**Code:**

#include<iostream>

using namespace std;

class employee{

int eId;

float salary;

char name[30];

public:

void getData()

{

cin>>eId>>name>>salary;

}

void display()

{

cout<<eId<<'\t'<<name<<'\t'<<salary<<'\t'<<"\n";

}

};

int main()

{

int i;

employee arr[3];

for(i=0;i<3;i++)

{

arr[i].getData();

}

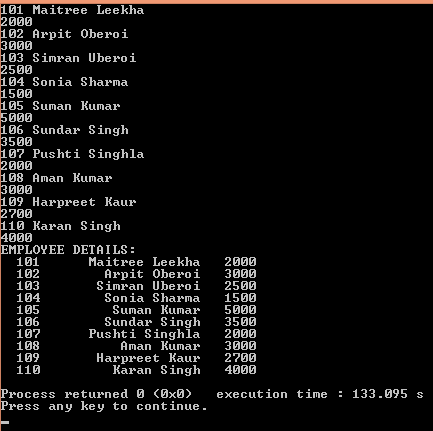
cout<<"EMPLOYEE DETAILS:\n";

for(i=0;i<3;i++)

{

arr[i].display();}}

**Result:**



**Discussion:**

A class is created wih three data members – id, name and salary. The gatData and displayData are defined.In the main program, an array of objects of the type employee is created and the values of these objects are input by the user as the getData function is called and are then printed using displayData function.

**Conclusion:**

By using array of objects we have successfully printed the employee details.