|  |  |
| --- | --- |
| **Group** | **A** |
| **Assignment No** | **01** |
| **Title** | In Second year Computer Engineering class of M students, set A of students play cricket and set B of students play badminton. Write C/C++ program to find and display-  i. Set of students who play either cricket or badminton or both  ii. Set of students who play both cricket and badminton  iii. Set of students who play only cricket  iv. Set of students who play only badminton  v. Number of students who play neither cricket nor badminton  (Note- While realizing the set duplicate entries are to avoided) |
| **Date of completion** | 29/06/2017 |
| **Marks out of 10** | 7 |
| **Signature of staff** |  |

**Assignment No:- A-1**

**Name: Ghodekar Prachi Shrikant.(Roll no. 12)**

**Objectives:**

1.To introduce the basics of theory of Sets and some of its applications.

2. To understand and implement the Array data structure

3. To understand how to pass arrays as parameters to functions.

**Problem Statement:**

In Second year Computer Engineering class of M students, set A of students play cricket and set B of students play badminton. Write C/C++ program to find and display-

1. Set of students who play either cricket or badminton or both
2. Set of students who play both cricket and badminton
3. Set of students who play only cricket
4. Set of students who play only badminton
5. Number of students who play neither cricket nor badminton

(Note- While realizing the set duplicate entries are to avoided)

**Problem Definition:**

**Sets :** A Set is a collection of well-defined objects. Sets are usually denoted by capital letters A, B, C, … etc.

e.g. A = { 1, 5, 8, 10 }

**Membership :** Objects that form a set are called members or elements of the set. An Object x is a member of set A, if x is one of the elements of set A. It is denoted by x  A.

Let A = { 1, 5, 8, 10 } then 5  A but 4  A.

**Equality of Sets :** If two sets A and B have the same elements, they are said to be equal sets. We write this as A = B.

Let A = { 1, 2 } B = { 1 , 2 }, Then A = B.

**Union of Sets :** Union of two sets A and B is a set of those elements which belong to A or to B or both A and B. It is denoted by A  B.

A  B = { x | x  A or x  B or x  both A and B }

A = { 1, 5, 8, 10 } B = { 3, 5, 10, 12 }

A  B = { 1, 3, 5, 8, 10, 12 }

**Intersection of Sets :** Intersection of two sets A and B is a set of those elements which belong to both A and B. It is denoted by A  B.

A  B = { x | x  A and x  B }

A = { 1, 5, 8, 10 } B = { 3, 5, 10, 12 }

A  B = { 5, 10 }

**Difference of Sets :** Difference of two sets A and B is a set of those elements which are present in A but not in B. It is denoted by A - B.

A – B = { x | x  A and x  B }

A = { 1, 5, 8, 10 } B = { 3, 5, 10, 12 }

A - B = { 1, 8 }

**Symmetric Difference :** Symmetric Difference of two sets A and B is a set of those elements that are present in A and B but not in both. It is denoted by A  B.

A  B = (A – B)  (B – A)

A = { 1, 5, 8, 10 } B = { 3, 5, 10, 12 }

A  B = { 1, 3 , 8 , 12 }

**Array :** An array is a collection of variables that are of the same data type. Each item in an array is called an element. All elements in an array are referenced by the name of the array and are stored in a set of consecutive memory slots.

Arrays are collection of homogeneous data types. As with most computers arrays provide a common name to the collection of that data type where each element can be referred to with an index number.

**Algorithms :**

Write down algorithms / Flowcharts for following modules

* Algorithm for accepting a Set
* Algorithm for Displaying a Set
* Algorithm for finding union of two Sets
* Algorithm for finding intersection of two Sets
* Algorithm for finding difference of two Sets

**Sample Input Output**

MENU

1. Accept Set
2. Display Set
3. Union
4. Intersection
5. Difference (A-B)
6. Difference (B-A)

**Accept a Set**

Enter the no. of elements in the array : 4

Enter the elements : Dhiraj Ajit Mukesh Sachin

**Display a Set**

A = { Dhiraj, Ajit, Mukesh, Sachin }

**Union**

A = { Dhiraj, Ajit, Mukesh, Sachin } B = { Suraj, Ajit, Sachin, Yogesh }

A  B = { Dhiraj, Suraj, Ajit, Mukesh, Sachin, Yogesh }

**Intersection**

A = { Dhiraj, Ajit, Mukesh, Sachin } B = { Suraj, Ajit, Sachin, Yogesh }

A  B = { Ajit, Sachin }

**Difference**

A = { Dhiraj, Ajit, Mukesh, Sachin } B = { Suraj, Ajit, Sachin, Yogesh }

A - B = { Dhiraj, Mukesh }

**Difference**

A = { Dhiraj, Ajit, Mukesh, Sachin } B = { Suraj, Ajit, Sachin, Yogesh }

B - A = { Suraj, Yogesh}

**Conclusion:**

Thus we have studied and implemented how to declare a 1D dimensional array, read the elements of the array and print the elements of the array.

We have also implemented the following operations on the arrays:

1. Insertion

2. Deletion

3. Updation

**Marks Obtained...........................**

**Sign of Teacher.............................**

**Code:**

#include<stdio.h>

int accept\_set(int A[])

{

int i,n;

printf("\nEnter the total no. of students in the Set : ");

scanf("%d",&n);

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

{

printf("Enter the %d element : ",i+1);

scanf("%d",&A[i]);

}

printf("\nSet accepted successfully\n");

return n;

}

void display\_set(int A[],int n)

{

int i;

printf("{");

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

{

printf("%d, ",A[i]);

}

printf("\b\b }\n");

}

int search\_set(int A[],int n,int X)

{

int i;

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

{

if(A[i] == X)

return 1;

}

return 0;

}

int find\_intersection(int A[],int na,int B[], int nb, int C[])

{

int nc = 0,i,flag;

int X;

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

{

X = A[i];

flag = search\_set(B,nb,X);

if(flag == 1)

{

C[nc] = X;

nc++;

}

}

return nc;

}

int find\_difference(int A[],int na,int B[], int nb, int C[])

{

int nc = 0,i,flag;

int X;

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

{

X = A[i];

flag = search\_set(B,nb,X);

if(flag == 0)

{

C[nc] = X;

nc++;

}

}

return nc;

}

int find\_union(int A[],int na,int B[], int nb, int C[])

{

int nc = 0,i,flag;

int X;

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

{

C[i] = A[i];

}

nc = na;

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

{

X = B[i];

flag = search\_set(A,na,X);

if(flag == 0)

{

C[nc] = X;

nc++;

}

}

return nc;

}

int main()

{

int ch,na,nb,nc,m;

int A[20],B[20],C[20];

//system("clear");

printf("\nEnter the total no. of students in the class : ");

scanf("%d",&m);

do

{

printf("\n\t\t\t1: Accept");

printf("\n\t\t\t2: Display");

printf("\n\t\t\t3: Union");

printf("\n\t\t\t4: Intersection");

printf("\n\t\t\t5: Difference A- B");

printf("\n\t\t\t6: Difference B -A");

printf("\n\t\t\t7: No. of students dont play cricket or badminton");

printf("\n\t\t\t8: Exit");

printf("\n\nEnter ur choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1 : printf("\nInput the Set A");

na = accept\_set(A);

printf("\nInput the Set B");

nb = accept\_set(B);

break;

case 2 : printf("\nSet A = ");

display\_set(A,na);

printf("\nSet B = ");

display\_set(B,nb);

break;

case 3 : printf("\nSet A = ");

display\_set(A,na);

printf("\nSet B = ");

display\_set(B,nb);

nc = find\_union(A,na,B,nb,C);

printf("\nSet A U B = ");

display\_set(C,nc);

break;

case 4 : printf("\nSet A = ");

display\_set(A,na);

printf("\nSet B = ");

display\_set(B,nb);

nc = find\_intersection(A,na,B,nb,C);

printf("\nSet A ^ B = ");

display\_set(C,nc);

break;

case 5 : printf("\nSet A = ");

display\_set(A,na);

printf("\nSet B = ");

display\_set(B,nb);

nc = find\_difference(A,na,B,nb,C);

printf("\nSet A - B = ");

display\_set(C,nc);

break;

case 6 : printf("\nSet A = ");

display\_set(A,na);

printf("\nSet B = ");

display\_set(B,nb);

nc = find\_difference(B,nb,A,na,C);

printf("\nSet B - A = ");

display\_set(C,nc);

break;

case 7 : printf("\nSet A = ");

display\_set(A,na);

printf("\nSet B = ");

display\_set(B,nb);

nc = find\_union(A,na,B,nb,C);

printf("\nNo of students who play neither cricket nor badminton = %d",m-nc);

break;

case 8 : printf("\nGoodbye\n");

break;

default:printf("\nInvalid choice !! Try again!!! \n\n");

}

}while(ch!= 8);

return 0;

}

**Output:**

/\*

Enter the total no. of students in the class : 50

1: Accept

2: Display

3: Union

4: Intersection

5: Difference A- B

6: Difference B -A

7: No. of students dont play cricket or badminton

8: Exit

Enter ur choice : 1

Input the Set A

Enter the total no. of students in the Set : 5

Enter the 1 element : 5

Enter the 2 element : 3

Enter the 3 element : 6

Enter the 4 element : 4

Enter the 5 element : 1

Set accepted successfully

Input the Set B

Enter the total no. of students in the Set : 5

Enter the 1 element : 1

Enter the 2 element : 3

Enter the 3 element : 4

Enter the 4 element : 5

Enter the 5 element : 6

Set accepted successfully

1: Accept

2: Display

3: Union

4: Intersection

5: Difference A- B

6: Difference B -A

7: No. of students dont play cricket or badminton

8: Exit

Enter ur choice : 2

Set A = {5, 3, 6, 4, 1 }

Set B = {1, 3, 4, 5, 6 }

1: Accept

2: Display

3: Union

4: Intersection

5: Difference A- B

6: Difference B -A

7: No. of students dont play cricket or badminton

8: Exit

Enter ur choice : 3

Set A = {5, 3, 6, 4, 1 }

Set B = {1, 3, 4, 5, 6 }

Set A U B = {5, 3, 6, 4, 1 }

1: Accept

2: Display

3: Union

4: Intersection

5: Difference A- B

6: Difference B -A

7: No. of students dont play cricket or badminton

8: Exit

Enter ur choice : 4

Set A = {5, 3, 6, 4, 1 }

Set B = {1, 3, 4, 5, 6 }

Set A ^ B = {5, 3, 6, 4, 1 }

1: Accept

2: Display

3: Union

4: Intersection

5: Difference A- B

6: Difference B -A

7: No. of students dont play cricket or badminton

8: Exit

Enter ur choice : 5

Set A = {5, 3, 6, 4, 1 }

Set B = {1, 3, 4, 5, 6 }

Set A - B = }

1: Accept

2: Display

3: Union

4: Intersection

5: Difference A- B

6: Difference B -A

7: No. of students dont play cricket or badminton

8: Exit

Enter ur choice : 6

Set A = {5, 3, 6, 4, 1 }

Set B = {1, 3, 4, 5, 6 }

Set B - A = }

1: Accept

2: Display

3: Union

4: Intersection

5: Difference A- B

6: Difference B -A

7: No. of students dont play cricket or badminton

8: Exit

Enter ur choice : 7

Set A = {5, 3, 6, 4, 1 }

Set B = {1, 3, 4, 5, 6 }

No of students who play neither cricket nor badminton = 45

1: Accept

2: Display

3: Union

4: Intersection

5: Difference A- B

6: Difference B -A

7: No. of students dont play cricket or badminton

8: Exit

Enter ur choice : 8

Goodbye

\*/

|  |  |
| --- | --- |
| **Group** | **A** |
| **Assignment No** | **02** |
| **Title** | Write C/C++ program to store marks scored for first test of subject 'Data Structures and Algorithms' for N students. Compute   1. The average score of class 2. Highest score and lowest score of class 3. Marks scored by most of the students 4. list of students who were absent for the test |
| **Date of completion** | 06/07/2017 |
| **Marks out of 10** | 8 |
| **Signature of Teacher** |  |

**Assignment No.A-2**

**Name: Ghodekar Prachi Shrikant.(Roll no. 12)**

**Objectives :-** To study Array operations for maintaining student database.

**Problem Statement:-**

Write C/C++ program to store marks scored for first test of subject 'Data Structures and Algorithms' for N students. Compute

1. The average score of class

2. Highest score and lowest score of class

3. Marks scored by most of the students

4. list of students who were absent for the test

**Software & Hardware requirements: -** 64-bit Open source Linux or its derivative , Open Source C++ Programming tool like G++/GCC

**THEORY-**

Array is a container which can hold fix number of items and these items should be of same type. Most of the data structure make use of array to implement their algorithms. Following are important terms to understand the concepts of Array.

* **Element** − Each item stored in an array is called an element.
* **Index** − Each location of an element in an array has a numerical index which is used to identify the element.

**Array Representation**

Arrays can be declared in various ways in different languages. For illustration, let's take C array declaration.

Arrays can be declared in various ways in different languages. For illustration, let's take C array declaration.

As per above shown illustration, following are the important points to be considered.

1. Index starts with 0.

2. Array length is 8 which means it can store 8 elements.

3. Each element can be accessed via its index. For example, we can fetch element at index 6 as 9.

**Basic Operations**

Following are the basic operations supported by an array.

1. **Traverse** − print all the array elements one by one.
2. **Insertion** − add an element at given index.
3. **Deletion** − delete an element at given index.
4. **Search** − search an element using given index or by value.
5. **Update** − update an element at given index.

In C, when an array is initialized with size, then it assigns defaults values to its elements in following order.

|  |  |
| --- | --- |
| **Data Type** | **Default Value** |
| Bool | false |
| Char | 0 |
| Int | 0 |
| Float | 0.0 |
| Double | 0.0f |

**Insertion Operation**

Insert operation is to insert one or more data elements into an array. Based on the requirement, new element can be added at the beginning, end or any given index of array.

Here, we see a practical implementation of insertion operation, where we add data at the end of the array −

Let Array is a linear unordered array of MAX elements.

**Result**

Let LA is a Linear Array (unordered) with N elements and K is a positive integer such that K<=N. Below is the algorithm where ITEM is inserted into the Kth position of LA −

1. Start

2. Set J=N

3. Set N = N+1

4. Repeat steps 5 and 6 while J >= K

5. Set LA[J+1] = LA[J]

6. Set J = J-1

7. Set LA[K] = ITEM

8. Stop

**Deletion Operation**

Deletion refers to removing an existing element from the array and re-organizing all elements of an array.

Consider LA is a linear array with N elements and K is a positive integer such that K<=N. Below is the algorithm to delete an element available at the Kth position of LA.

1. Start

2. Set J=K

3. Repeat steps 4 and 5 while J < N

4. Set LA[J-1] = LA[J]

5. Set J = J+1

6. Set N = N-1

7. Stop

**Search Operation**

You can perform a search for array element based on its value or its index.

Consider LA is a linear array with N elements and K is a positive integer such that K<=N. Below is the algorithm to find an element with a value of ITEM using sequential search.

1. Start

2. Set J=0

3. Repeat steps 4 and 5 while J < N

4. IF LA[J] is equal ITEM THEN GOTO STEP 6

5. Set J = J +1

6. PRINT J, ITEM

7. Stop

**Update Operation**

Update operation refers to updating an existing element from the array at a given index.Consider LA is a linear array with N elements and K is a positive integer such that K<=N. Below is the algorithm to update an element available at the Kth position of LA.

1. Start

2. Set LA[K-1] = ITEM

3. Stop

**Algorithm:-**

1. Accept number of students and their marks as an array.
2. Print menu for average, minimum, maximum marks and for counting maximum number of students marks.
3. Accept choice
4. If choice is one calculate average as

Avg=total/no of students and print average

5. If choice is two

a)find minimum and maximum marks as

Initially min=max=marks[0] i.e first student marks as min and max.

b) Compare with this if marks<min then update min.

c)Similarly if marks>max update max

6. If choice is three compare array element and count frequency of elements

From that print element having heighest frequency

**Conclusion:**

Thus we have studied and implemented how to declare a 1D dimensional array, read the elements of the array and print the elements of the array.

We have also implemented the following operations on the arrays:

1. Insertion

2. Deletion

3. Updation

**Marks Obtained...........................**

**Sign of Teacher.............................**

**Code:**

#include<iostream>

using namespace std;

#define MAX 60

int avg(int marks[],int n);

int maxmin(int marks[],int n);

int occure(int marks[],int n);

int main()

{

int marks[MAX];

int i,n,choice;

cout<<"\n Enter the no. of Students in SE Comp:"<<endl;

cin>>n;

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

{

cout<<"\n Enter the marks for Roll no.:"<<endl;

cin>>marks[i];

}

//for(i=0;i<n;i++)

{

cout<<"\n Roll no DSA marks:"<<marks[i];

}

do

{

cout<<"\n 1.Average score of class \n 2.Maximum & Minimum \n 3.Marks get by most of the student \n 4.Exit"<<endl;

cout<<"\n Enter your choice"<<endl;

cin>>choice;

switch(choice)

{

case 1: avg(marks,n);

break;

case 2: maxmin(marks,n);

break;

case 3: occure(marks,n);

break;

}

}while(choice!=4);

}

int avg(int marks[],int n)

{

float total=0;

int i;

float average;

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

{

total=total+marks[i];

}

average=total/n;

cout<<"\n Average marks of students"<<average<<endl;

return average;

}

int maxmin(int marks[],int n)

{

int min,max,i;

min=max=marks[0];

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

{

if(marks[i]<min)

min=marks[i];

}

cout<<"\n Minimum marks are"<<endl;

cout<<min;

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

{

if(marks[i]>max)

max=marks[i];

}

cout<<"\n Maximum marks are"<<endl;

cout<<max;

return min;

}

int occure(int marks[],int n)

{

int count,i,j,max;

int number=1;

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

{

count=1;

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

{

if(marks[i]==marks[j])

count++;

}

if(count>max)

{

max=count;

number=marks[i];

}

}

cout<<"\n Student got the marks"<<max<<number;

return 1;

}

**Output:**

/\*

Enter the no. of Students in SE Comp:

7

Enter the marks for Roll no.:

44

Enter the marks for Roll no.:

50

Enter the marks for Roll no.:

52

Enter the marks for Roll no.:

62

Enter the marks for Roll no.:

63

Enter the marks for Roll no.:

63

Enter the marks for Roll no.:

89

Roll no DSA marks:49

1.Average score of class

2.Maximum & Minimum

3.Marks get by most of the student

4.Exit

Enter your choice

1

Average marks of students60.4286

1.Average score of class

2.Maximum & Minimum

3.Marks get by most of the student

4.Exit

Enter your choice

2

Minimum marks are

44

Maximum marks are

89

1.Average score of class

2.Maximum & Minimum

3.Marks get by most of the student

4.Exit

Enter your choice

3

Student got the marks63000961

1.Average score of class

2.Maximum & Minimum

3.Marks get by most of the student

4.Exit

Enter your choice

4

\*/

|  |  |
| --- | --- |
| **Group** | **A** |
| **Assignment No** | **03** |
| **Title** | Write C++ program for sparse matrix realization and operations on it- Transpose, Fast Transpose and addition of two matrices. |
| **Date of completion** | 13/07/2017 |
| **Marks out of 10** | 7 |
| **Signature of Teacher** |  |

**Assignment No.A-3**

**Name: Ghodekar Prachi Shrikant.(Roll no. 12)**

**Objectives:**

1. To explain the efficient usage of arrays in defining and implementing Sparse Matrices.
2. To implement operations on Sparse Matrices.

**Problem Statement :**

Write C++ program for sparse matrix realization and operations on it- Transpose, Fast Transpose and addition of two matrices.

**Problem Definition :**

**Definition of Sparse Matrix :** A Sparse matrix is a matrix having very few non-zero elements and a large number of elements are Zero.

Example

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | 10 | 0 | 0 | 0 | -5 | 0 | 0 | 0 |  |
|  | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |  |
| **M =** | 0 | 0 | 0 | 0 | -4 | 0 | 0 | 0 |  |
|  | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
|  | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

In this example 6 of the 40 elements are non-zero. Hence it can be called a sparse matrix. Such a matrix does not utilize memory efficiently. Hence, a different representation and handling is developed for sparse matrix.

A Sparse matrix can be stored as a list of tuples storing only the non-zero elements and their row and column number. Thus the above matrix will be stored as:

|  |  |  |
| --- | --- | --- |
| **Row No** | **Column No** | **Value** |
| **5** | **8** | **6** |
| 0 | 0 | 10 |
| 0 | 4 | -5 |
| 1 | 2 | 1 |
| 2 | 4 | -4 |
| 3 | 0 | 7 |
| 4 | 1 | 9 |

The first tuple gives the information about the matrix namely number of rows, number of columns and the number of non-zero elements. The remaining give information about each non-zero value.

Matrices play a very important role for solving many problems in scientific applications. Therefore, they have to be efficiently represented such that the operations on matrices like addition, multiplication, inverse, transpose etc. can be carried out with minimum time and space requirements.

**Addition of Sparse Matrix**

Two sparse matrices can be added only if their orders are same.

Elements in the same position (having same row and column number) are to be added and stored in the resultant matrix. Otherwise both elements have to be separately copied in the resultant matrix.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Row No** | **Column No** | **Value** |  |  | **Row No** | **Column No** | **Value** |
| **5** | **8** | **6** |  |  | **5** | **8** | **5** |
| 0 | 0 | 10 | + | | 0 | 1 | 2 |
| 0 | 4 | -5 | 0 | 4 | 7 |
| 1 | 2 | 1 | 1 | 1 | 3 |
| 2 | 4 | -4 | 1 | 2 | 2 |
| 3 | 0 | 7 |  |  | 3 | 0 | 4 |
| 4 | 1 | 9 |  |  |  |  |  |

**Addition Result**

|  |  |  |
| --- | --- | --- |
| **Row No** | **Column No** | **Value** |
| **5** | **8** | **8** |
| 0 | 0 | 10 |
| 0 | 1 | 2 |
| 0 | 4 | 2 |
| 1 | 1 | 3 |
| 1 | 2 | 3 |
| 2 | 4 | -4 |
| 3 | 0 | 11 |
| 4 | 1 | 9 |

**Simple Transpose of a Sparse Matrix**

A transpose of a matrix is a matrix which contains the i, jth element of the original matrix in the j, ith position of the transpose i.e T[i][j] = A[j][i] .

**Sparse Matrix A**

|  |  |  |
| --- | --- | --- |
| **Row No** | **Column No** | **Value** |
| **5** | **8** | **6** |
| 0 | 0 | 10 |
| 0 | 4 | -5 |
| 1 | 2 | 1 |
| 2 | 4 | -4 |
| 3 | 0 | 7 |
| 4 | 1 | 9 |

For the above sparse matrix, its transpose will be

**Transpose Matrix**

|  |  |  |
| --- | --- | --- |
| **Row No** | **Column No** | **Value** |
| **5** | **8** | **6** |
| 0 | 0 | 10 |
| 0 | 3 | 7 |
| 1 | 4 | 9 |
| 2 | 1 | 1 |
| 4 | 0 | -5 |
| 4 | 2 | -4 |

If we simply interchange the row and column numbers, the elements will not be in the proper order i.e. the element A (row,col,value) will be at T(col, row,value) but we do not know where to put the element.

If we store them consecutively, we will have to move tuples every time a new tuple has to be inserted to maintain the sequence.

This can be avoided by finding all elements in column i of A and put them in row i of T for all 0 ≤ i < num – cols.

i.e. for all columns j in matrix A do

for all elements in matrix A do

place A(i , j, value) in T(j,i, value)

**Analysis:**

If there are n columns in matrix A of order m x n and the number of non-zero terms is t, the computing time is O(nt).

If t is the order of nm, the time for this algorithm becomes O(n2m) which is far worse than O(nm) time for transpose of normal matrices.

**Fast Transpose**

This method achieves the transpose of a sparse matrix in O(n+t) time and hence is much faster than the simple transpose method. This method is as follows

1. Find the number of elements in each column of A
2. This gives the number of elements in each row of T
3. Using this, calculate the starting point of each row of T.
4. Move each element of A one by one to its correct position in T.

Example :

**Sparse Matrix A**

|  |  |  |
| --- | --- | --- |
| **Row No** | **Column No** | **Value** |
| **5** | **8** | **6** |
| 0 | 0 | 10 |
| 0 | 4 | -5 |
| 1 | 2 | 1 |
| 2 | 4 | -4 |
| 3 | 0 | 7 |
| 4 | 1 | 9 |

Number of elements in each column of A

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Count** | 2 | 1 | 1 | 0 | 2 | 0 | 0 | 0 |

The starting positions of each row in T.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Position** | 0 | 2 | 3 | 4 | 4 | 6 | 6 | 6 |

Sparse Matrix A Transpose

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Row No** | **Column No** | **Value** |  |  | **Row No** | **Column No** | **Value** |
| **5** | **8** | **6** |  |  | **5** | **8** | **6** |
| 0 | 0 | 10 |  |  | 0 | 0 | 10 |
| 0 | 4 | -5 |  |  | 0 | 3 | 7 |
| 1 | 2 | 1 |  |  | 1 | 4 | 9 |
| 2 | 4 | -4 |  |  | 2 | 1 | 1 |
| 3 | 0 | 7 |  |  | 4 | 0 | -5 |
| 4 | 1 | 9 |  |  | 4 | 2 | -4 |

**Analysis of Fast Transpose :**

1. The steps taken to find number of elements in each column of A=n [Assuming a m x n matrix]
2. The steps taken to compute the starting positions of each row in T = n -1
3. The elements of A are copied to T in t steps

Therefore, Total no. of steps = n + n – 1 + t

Which is of the order O(n+t).

If t is of the order nm, this method becomes O(nm) which is same for two dimensional arrays.

**Algorithms :**

Write algorithms for following modules:

1. Algorithm for Sparse Matrix addition

2. Algorithm for Simple Transpose

3. Algorithm for Fast Transpose

**Sample Input Output:**

**MENU**

1. Accept the matrix and convert it into sparse Matrix Representation
2. Simple Transpose
3. Fast Transpose
4. Addition of two Sparse matrix

**Input :** Enter the order of Matrix : 5 8

Enter the Matrix

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  | 10 | 0 | 0 | 0 | -5 | 0 | 0 | 0 |  |
|  | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |  |
| M = | 0 | 0 | 0 | 0 | -4 | 0 | 0 | 0 |  |
|  | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
|  | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |

**Output :**

Sparse Matrix Representation :

|  |  |  |
| --- | --- | --- |
| **Row No** | **Column No** | **Value** |
| **5** | **8** | **6** |
| 0 | 0 | 10 |
| 0 | 4 | -5 |
| 1 | 2 | 1 |
| 2 | 4 | -4 |
| 3 | 0 | 7 |
| 4 | 1 | 9 |

**Simple Transpose:**

Sparse Matrix A Transpose

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Row No** | **Column No** | **Value** |  |  | **Row No** | **Column No** | **Value** |
| **5** | **8** | **6** |  |  | **5** | **8** | **6** |
| 0 | 0 | 10 |  |  | 0 | 0 | 10 |
| 0 | 4 | -5 |  |  | 0 | 3 | 7 |
| 1 | 2 | 1 |  |  | 1 | 4 | 9 |
| 2 | 4 | -4 |  |  | 2 | 1 | 1 |
| 3 | 0 | 7 |  |  | 4 | 0 | -5 |
| 4 | 1 | 9 |  |  | 4 | 2 | -4 |

**Fast Transpose :**

Sparse Matrix A Transpose

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Row No** | **Column No** | **Value** |  |  | **Row No** | **Column No** | **Value** |
| **5** | **8** | **6** |  |  | **5** | **8** | **6** |
| 0 | 0 | 10 |  |  | 0 | 0 | 10 |
| 0 | 4 | -5 |  |  | 0 | 3 | 7 |
| 1 | 2 | 1 |  |  | 1 | 4 | 9 |
| 2 | 4 | -4 |  |  | 2 | 1 | 1 |
| 3 | 0 | 7 |  |  | 4 | 0 | -5 |
| 4 | 1 | 9 |  |  | 4 | 2 | -4 |

**Addition:**

Sparse Matrix A Sparse Matrix B

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Row No** | **Column No** | **Value** |  |  | **Row No** | **Column No** | **Value** |
| **5** | **8** | **6** |  |  | **5** | **8** | **5** |
| 0 | 0 | 10 | + | | 0 | 1 | 2 |
| 0 | 4 | -5 | 0 | 4 | 7 |
| 1 | 2 | 1 | 1 | 1 | 3 |
| 2 | 4 | -4 | 1 | 2 | 2 |
| 3 | 0 | 7 |  |  | 3 | 0 | 4 |
| 4 | 1 | 9 |  |  |  |  |  |

**Addition Result**

|  |  |  |
| --- | --- | --- |
| **Row No** | **Column No** | **Value** |
| **5** | **8** | **8** |
| 0 | 0 | 10 |
| 0 | 1 | 2 |
| 0 | 4 | 2 |
| 1 | 1 | 3 |
| 1 | 2 | 3 |
| 2 | 4 | -4 |
| 3 | 0 | 11 |
| 4 | 1 | 9 |

**Conclusion:**

Thus we have represented Sparse matrix using array and performed matrix addition, simple and fast transpose

**Design Experiments:**

* 1. Perform the above assignment by dynamically allocating memory for sparse matrix representation and using pointers perform the operations like addition, simple and fast transpose.
  2. Perform multiplication of two sparse matrix

**Oral Questions:**

1. What is a sparse matrix?
2. What is the advantage of sparse matrix?
3. Explain the analysis of Simple Transpose?
4. Explain the analysis of Fast Transpose?
5. How sparse matrix can be represented using a 2 D array?
6. How sparse matrix can be represented using structures?
7. What are the applications of Sparse Matrix?
8. How to represent sparse matrix using link list?
9. Compare the time complexity of Matrix addition using normal matrices and Sparse Matrix representation?

**Marks obtained......................**

**Sign of Teacher.......................**

**Code:**

#include<iostream>

#include<iomanip>

#include<stdlib.h>

using namespace std;

struct SPARSE

{

int row;

int col;

int val;

};

typedef struct SPARSE mysparse;

void accept\_matrix(int A[8][8],int \*rp,int \*cp)

{

int i,j,r,c;

cout<<"\nEnter the order of Matrix (r,c) : ";

cin>>r>>c;

cout<<"\nEnter the elements of the matrix\n";

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

{

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

{

cin>>A[i][j];

}

}

\*rp = r;

\*cp = c;

cout<<"\nSparse Matrix accepted successfully\n";

}

void display\_matrix(int A[8][8],int r,int c)

{

int i,j;

cout<<"Matrix ("<<r<<","<<c<<") : ";

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

{

cout<<"\n\t\t";

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

{

cout<<setw(5)<<A[i][j];

}

}

}

void convert\_to\_sparse\_representation(int A[8][8],int r,int c, mysparse S[])

{

int i,j,k=1;

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

{

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

{

if(A[i][j] != 0)

{

S[k].row = i;

S[k].col = j;

S[k].val = A[i][j];

k++;

}

}

}

S[0].row = r;

S[0].col = c;

S[0].val = k-1;

cout<<"\nMatrix converted into sparse representation\n";

}

void display\_sparse\_matrix(mysparse S[])

{

int i;

cout<<"Sparse Matrix is ";

cout<<"\n\t Row Col Val";

cout<<"\n\t-----------------";

cout<<"\n\t"<<setw(4)<<S[0].row<<setw(6)<<S[0].col<<setw(6)<<S[0].val;

cout<<"\n\t-----------------";

for( i = 1 ; i <= S[0].val ; i++ )

{

cout<<"\n\t"<<setw(4)<<S[i].row<<setw(6)<<S[i].col<<setw(6)<<S[i].val;

}

cout<<"\n\t-----------------\n";

}

void addition\_of\_sparse\_matrix(mysparse S1[],mysparse S2[],mysparse S3[])

{

int i,j,k,t1 = S1[0].val,t2 = S2[0].val;

int flag1[10]={0};

int flag2[10]={0};

i = j = k = 1;

if(S1[0].row==S2[0].row && S1[0].col==S2[0].col)

{

for(i=1;i<=t1;i++)

{

for(j=1; j<=t2;j++)

{

if(S1[i].row == S2[j].row && S1[i].col == S2[j].col && flag1[i]==0 && flag2[j]==0)

{

S3[k].row = S1[i].row;

S3[k].col = S1[i].col;

S3[k].val = S1[i].val + S2[j].val;

k++;

flag1[i]=1;

flag2[j]=1;

}

}

}

for(i=1;i<=t1;i++)

{

if(flag1[i]!=1)

{

S3[k].row = S1[i].row;

S3[k].col = S1[i].col;

S3[k].val = S1[i].val;

k++;

}

}

for(i=1;i<=t2;i++)

{

if(flag2[i]!=1)

{

S3[k].row = S2[i].row;

S3[k].col = S2[i].col;

S3[k].val = S2[i].val;

k++;

}

}

S3[0].row = S1[0].row;

S3[0].col = S1[0].col;

S3[0].val = k-1;

cout<<"\nAddition done successfully\n";

}

else

cout<< " Addition not Possible ";

}

void simple\_transpose\_of\_sparse\_matrix(mysparse S[],mysparse T[])

{

int k = 1,c,i;

for(c = 0; c < S[0].col; c++)

{

for(i = 1; i<= S[0].val; i++)

{

if(S[i].col == c)

{

T[k].row = S[i].col;

T[k].col = S[i].row;

T[k].val = S[i].val;

k++;

}

}

}

T[0].row = S[0].col;

T[0].col = S[0].row;

T[0].val = S[0].val;

cout<<"\nSimple Transpose done successfully\n";

}

void fast\_transpose\_of\_sparse\_matrix(mysparse S[],mysparse T[])

{

int k = 1,c,i;

int Count[15],Pos[15];

for(i=0;i<S[0].col;i++)

Count[i] = 0;

for(i = 1; i<= S[0].val; i++)

{

c = S[i].col;

Count[c]++;

}

Pos[0] = 1;

for(c = 1; c < S[0].col; c++)

{

Pos[c] = Pos[c - 1] + Count[c-1];

}

for(i = 1; i<= S[0].val; i++)

{

c = S[i].col;

k = Pos[c];

T[k].row = S[i].col;

T[k].col = S[i].row;

T[k].val = S[i].val;

Pos[c]++;

}

T[0].row = S[0].col;

T[0].col = S[0].row;

T[0].val = S[0].val;

cout<<"\nFast Transpose done successfully\n";

}

int main()

{

int ch;

int M1[8][8],r1,c1;

mysparse S1[15],S2[15],Ans[15];

do

{

cout<<"\n\t\t1: Accept and convert to Sparse representation";

cout<<"\n\t\t2: Addition ";

cout<<"\n\t\t3: Simple Transpose";

cout<<"\n\t\t4: Fast Transpose";

cout<<"\n\t\t5: Exit";

cout<<"\n\nEnter your choice : ";

cin>>ch;

switch(ch)

{

case 1 : cout<<"\nInput First Sparse Matrix";

accept\_matrix(M1,&r1,&c1);

cout<<"\nFirst ";

display\_matrix(M1,r1,c1);

convert\_to\_sparse\_representation(M1,r1,c1,S1);

cout<<"\nFirst ";

display\_sparse\_matrix(S1);

cout<<"\nInput Second Sparse Matrix";

accept\_matrix(M1,&r1,&c1);

cout<<"\nSecond ";

display\_matrix(M1,r1,c1);

convert\_to\_sparse\_representation(M1,r1,c1,S2);

cout<<"\nSecond ";

display\_sparse\_matrix(S2);

break;

case 2 : cout<<"\nFirst ";

display\_sparse\_matrix(S1);

cout<<"\nSecond ";

display\_sparse\_matrix(S2);

addition\_of\_sparse\_matrix(S1,S2,Ans);

cout<<"\nAddition Resultant ";

display\_sparse\_matrix(Ans);

break;

case 3 : cout<<"\nFirst ";

display\_sparse\_matrix(S1);

simple\_transpose\_of\_sparse\_matrix(S1,Ans);

cout<<"\nTranspose Resultant ";

display\_sparse\_matrix(Ans);

break;

case 4 : cout<<"\nSecond ";

display\_sparse\_matrix(S2);

fast\_transpose\_of\_sparse\_matrix(S2,Ans);

cout<<"\nTranspose Resultant ";

display\_sparse\_matrix(Ans);

break;

case 5 : cout<<"\nGood Bye\n";

break;

default: cout<<"\nInvalid choice !! Try again\n";

}

}while(ch!=5);

return 0;

}

**Output:**

/\*

1: Accept and convert to Sparse representation

2: Addition

3: Simple Transpose

4: Fast Transpose

5: Exit

Enter your choice : 1

Input First Sparse Matrix

Enter the order of Matrix (r,c) : 3

3

Enter the elements of the matrix

5

4

2

4

5

6

1

2

3

Sparse Matrix accepted successfully

First Matrix (3,3) :

5 4 2

4 5 6

1 2 3

Matrix converted into sparse representation

First Sparse Matrix is

Row Col Val

-----------------

3 3 9

-----------------

0 0 5

0 1 4

0 2 2

1 0 4

1 1 5

1 2 6

2 0 1

2 1 2

2 2 3

-----------------

Input Second Sparse Matrix

Enter the order of Matrix (r,c) : 3

3

Enter the elements of the matrix

3

2

4

6

5

1

2

4

1

Sparse Matrix accepted successfully

Second Matrix (3,3) :

3 2 4

6 5 1

2 4 1

Matrix converted into sparse representation

Second Sparse Matrix is

Row Col Val

-----------------

3 3 9

-----------------

0 0 3

0 1 2

0 2 4

1 0 6

1 1 5

1 2 1

2 0 2

2 1 4

2 2 1

-----------------

1: Accept and convert to Sparse representation

2: Addition

3: Simple Transpose

4: Fast Transpose

5: Exit

Enter your choice : 2

First Sparse Matrix is

Row Col Val

-----------------

3 3 9

-----------------

0 0 5

0 1 4

0 2 2

1 0 4

1 1 5

1 2 6

2 0 1

2 1 2

2 2 3

-----------------

Second Sparse Matrix is

Row Col Val

-----------------

3 3 9

-----------------

0 0 3

0 1 2

0 2 4

1 0 6

1 1 5

1 2 1

2 0 2

2 1 4

2 2 1

-----------------

Addition done successfully

Addition Resultant Sparse Matrix is

Row Col Val

-----------------

3 3 9

-----------------

0 0 8

0 1 6

0 2 6

1 0 10

1 1 10

1 2 7

2 0 3

2 1 6

2 2 4

-----------------

1: Accept and convert to Sparse representation

2: Addition

3: Simple Transpose

4: Fast Transpose

5: Exit

Enter your choice : 3

First Sparse Matrix is

Row Col Val

-----------------

3 3 9

-----------------

0 0 5

0 1 4

0 2 2

1 0 4

1 1 5

1 2 6

2 0 1

2 1 2

2 2 3

-----------------

Simple Transpose done successfully

Transpose Resultant Sparse Matrix is

Row Col Val

-----------------

3 3 9

-----------------

0 0 5

0 1 4

0 2 1

1 0 4

1 1 5

1 2 2

2 0 2

2 1 6

2 2 3

-----------------

1: Accept and convert to Sparse representation

2: Addition

3: Simple Transpose

4: Fast Transpose

5: Exit

Enter your choice : 4

Second Sparse Matrix is

Row Col Val

-----------------

3 3 9

-----------------

0 0 3

0 1 2

0 2 4

1 0 6

1 1 5

1 2 1

2 0 2

2 1 4

2 2 1

-----------------

Fast Transpose done successfully

Transpose Resultant Sparse Matrix is

Row Col Val

-----------------

3 3 9

-----------------

0 0 3

0 1 6

0 2 2

1 0 2

1 1 5

1 2 4

2 0 4

2 1 1

2 2 1

-----------------

1: Accept and convert to Sparse representation

2: Addition

3: Simple Transpose

4: Fast Transpose

5: Exit

Enter your choice : 5

Good Bye

\*/

|  |  |
| --- | --- |
| **Group** | **A** |
| **Assignment No** | **04** |
| **Title** | Write C++ program for string operations- copy, concatenate, check substring, equal, reverse and length |
| **Date of completion** | 20/07/2017 |
| **Marks out of 10** | 8 |
| **Signature of staff** |  |

**Assignment No:- A-4**

**Name: Ghodekar Prachi Shrikant.(Roll no. 12)**

**Objectives:**

1. 1. To implement different operations on strings like Creating, copying, modifying, concatenating, reversing , Finding substrings, etc.
2. 2. To simulate the inbuilt string operations function.

**Problem Statement :**

Write C++ program for string operations- copy, concatenate, check substring, equal, reverse and length.

**Problem Definition :**

**String :** A string is a character array terminated by a null character (\0).

In C, the null character can be used to mark the end of a string. A string constant is a series of characters enclosed by double quotes. The C compiler automatically appends a null character to the array that has been initialized by a string constant.

char str[7] = "Hello!";

**Substring :** A string is a substring of a main string if it is a part of the main string.

**Palindrome :** A string is a palindrome if reverse of the string is equal to the original string.

e.g : nitin

**Different string operations are as follows :**

1. int strlen( const char \*str) -- Calculates string length excluding last null character.
2. char \* strcpy( char \*dest, const char \* src) – Copies string from source to destination
3. char \*strcat(char \*dest, const char \* src ) – Appends source string at the end of destination and returns pointer to destination
4. int strcmp(const char \*str1, const char \* str2) – Does unsigned comparison of two strings character by character and returns difference as integer.

If diff = 0 strings are equal

If diff < 0 string1 is smaller than string2

If diff > 0 string1 is greater than string2

1. char \* strrev( char \*str) – Reverses the input string and returns pointer to reversed

string.

1. char \*strstr( char \*str1, char \*str2) – Checks for string2 in string1 and returns pointer to location of first occurrence.
2. String palindrome – Checking if reversed string is same as original string.

**The strlen() Function**

The strlen() function can be used to measure the length of a string. This function does not count the null character in the last element

The syntax for the strlen() function is

size\_t strlen(const char \*s);

Here s is a char pointer variable. The return value from the function is the number of bytes. size\_t is a data type defined in the string.h header file. The size of the data type depends on the particular computer system.

**The strcpy() Function**

If you want to copy a string from one array to another, you can copy each item of the first array to the corresponding element in the second array, or you can simply call the C function strcpy() to do the job for you.

The syntax for the strcpy() function is

char \*strcpy(char \*dest, const char \*src);

Here the content of the string src is copied to the array referenced by dest. The strcpy() function returns the value of src if it is successful. The header file string.h must be included in your program before the strcpy() function is called.

**The strcat() Function**

strcat appends a copy of src to the end of dest. The length of the resulting string is strlen(dest) + strlen(src).

The syntax for the strcat() function is

char \*strcat(char \*dest, const char \*src);

strcat returns a pointer to the concatenated strings.

**The strrev() Function**

Reverses all characters in a string (except for the terminating null)

The syntax for the strrev() function is

char \*strrev(char \*s);

For example, it would change string\0 to gnirts\0

strrev returns a pointer to the reversed string.

**The strcmp() Function**

Compares two strings. The string comparison starts with the first character in each string and continues with subsequent characters until the corresponding characters differ or until the end of the strings is reached.

The syntax for the strcmp() function is

int strcmp(const char \*s1, const char\*s2);

This function returns an int value that is

< 0 if s1 < s2

== 0 if s1 == s2

> 0 if s1 > s2

**The strlwr() Function**

Converts uppercase letters (A to Z) in string s to lowercase (a to z).

The syntax for the strlwr() function is

char \*strlwr(char \*s);

No other characters are changed. Return Value is a pointer to the string s.

**The strupr() Function**

Converts lowercase letters (a to z) in string s to uppercase (A to Z).

The syntax for the strupr() function is

char \*strupr(char \*s);

No other characters are changed. Return Value is a pointer to the string s.

**The Strstr() Function**

Finds the first occurrence of a substring in another string

The syntax for the strupr() function is

char \*strstr(const char \*s1, const char \*s2);

strstr scans s1 for the first occurrence of the substring s2.

Return Value:

On success, strstr returns a pointer to the element in s1 where s2 begins (points to s2 in s1).

On error (if s2 does not occur in s1), strstr returns null.

**Algorithms :**

Write algorithms for following modules:

1. Algorithm for finding the string length

2. Algorithm for String Copy operation

3. Algorithm for Sting concatenation Operation

4. Algorithm for String Reverse Operation

5. Algorithm for String compare operations

6. Algorithm for substring operations

**Sample Input Output**

**MENU FOR STRING OPERATIONS**

1. String Length
2. String Copy
3. String Concatenation
4. String Reverse
5. String Compare
6. Substring

**Input :**

**Let**  String 1 = “ Fundamentals” String 2 = “ment”

Choose one of the operations to be performed

**Output :**

1. StringLength (string1 )

Length of string1 i.e 12 will be returned

1. StringCopy (string1,string2)

String 2 will also consist “ Fundamentals”

1. StringConcat (string1,string2)

String2 will get appended to string1

String2 will have “ FundamentalsFundamentals”

1. StringReverse ( string1,string2)

String2 will consist reverse of string1 i.e. “slatnemadnuF”

1. StringCompare ( string1,string2)

String 1 is greater than string 2

1. SubString (string1,string2)

If string1 = “ Fundamentals” and string2 =”ment”

String 2 is the substring of string1.

**Conclusion:**

Thus the String Handling Functions (StringCopy , StringLength, StringCompare , StringReverse, StringConcat , SubString) have been implemented (without using standard library routines).

**Design Experiments:**

* 1. **Simulate the following string library functions with pointers to arrays.**

char \*strncpy(s,ct,n) Copies at most n characters of string ct

to s Returns s. Pads with '\0's if

t has fewer than n characters.

char \*strncat(s,ct,n) Concatenates at most n characters of string

ct to end of string s; terminates s with

'\0'. Returns s.

int \*strncmp(cs,ct,n) Compares at most n characters of string cs

to string ct.

Returns < 0 if cs < ct

0 if cs= =ct

or > 0 if cs > ct

char \*strchr(cs,c) Returns a pointer to the first occurrence

of c in cs or NULL if not present.

char \*strrchr(cs,c) Returns a pointer to the last occurrence

of c in cs or NULL if not present.

* 1. **Simulate the following string library functions with pointers to arrays**

size\_t strspn(cs,ct) Returns length of prefix of cs consisting

of characters in ct.

size\_t strcspn(cs,ct) Returns length of prefix of cs consisting

of characters not in ct.

char \*strpbrk(cs,ct) Returns pointer to first occurrence in string

cs of any character of string ct, or NULL if

none present.

char \*strstr(cs,ct) Returns pointer to first occurrence of string

ct in cs, or NULL if not present.

size\_t strlen(cs) Returns the length of string cs.

char \*strerror(n) Returns pointer to implementation-defined

string corresponding to error n.

char \*strtok(s,ct) strtok searches s for tokens delimited by

characters from ct, NULL if none are found.

**Oral Questions:**

1. What is a string? How do you know its length?
2. What are the main differences between a string constant and a character constant?
3. Does the gets() function save the newline character from the standard input stream?
4. What types of data can the scanf() function read?
5. What are the left and right values (lvalue & rvalue)?
6. How can you obtain the address of a variable?
7. What is the concept of indirection in terms of using pointers?
8. Can a null pointer point to valid data?
9. What is the main difference between using scanf & gets for accepting strings ?
10. How do you reference an array by using a pointer?

**Marks Obtained........................**

**Sign of Teacher..........................**

**Code:**

#include<iostream>

#include<stdlib.h>

#include<cstring>

using namespace std;

int main()

{

int ch;

char str1[20];

char str2[30];

char str3[30];

int len;

//system("clear");

cout<<"\nEnter the source string : ";

cin>>str1;

cout<<"\nSource string is "<<str1;

do

{

cout<<"\n\t\t\t1: String length";

cout<<"\n\t\t\t2: String Copy";

cout<<"\n\t\t\t3: String Concat";

cout<<"\n\t\t\t4: String Reverse";

cout<<"\n\t\t\t5: String compare";

cout<<"\n\t\t\t6: Substring";

cout<<"\n\t\t\t7: Exit";

cout<<"\n\nEnter ur choice : ";

cin>>ch;

switch(ch)

{

case 1 : cout<<"\nSource string is "<<str1;

len = strlen(str1);

cout<<"\nLength of the string is "<<len;

break;

case 2 : cout<<"\nSource string is "<<str1;

//mystrcpy(D,S);

cout<<"\nCopied string is "<<strcpy(str2,str1);

break;

case 3 : cout<<"\nEnter the destination string : ";

cin>>str3;

cout<<"\nSource string is "<<str1;

cout<<"\nDestination string is "<<str3;

//mystrcat(D,S);

cout<<"\nConcatinated string is "<<strcat(str3,str1);

break;

case 4 : cout<<"\nSource string is "<<str1;

//mystrrev(S);

//cout<<"\nReverse string is "<<strrev(str1);

break;

case 5 : cout<<"\nEnter the first string : ";/\*

cin>>str1;

cout<<"\nEnter the second string : ";

cin>>str2;

cout<<"\nFirst string is "<<str1;

cout<<"\nSecond string is "<<str2;

len = strcmp(str1,str2);

if(len == 0)

cout<<"\nBoth strings are equal";

else

{

if( len > 0)

cout<<"\nFirst string is greater than second string";

else

cout<<"\nSecond string is greater than first string";

}

break;

case 6 : cout<<"\nEnter the Main string : ";

cin>>str1;

cout<<"\nEnter the string to be checked-substring : ";

cin>>str3;

cout<<"\nMain string is "<<str1;

cout<<"\nSecond string is "<<str3;

//len = strstr(S,D);

if(len == 0)

cout<<"\nSecond is not a substring of main string";

else

cout<<"\nSecond string is a substring of first string with "<<len<<" occurences";

break;

case 7 : cout<<"\nGoodbye\n";

break;

default:cout<<"\nInvalid choice !! Try again!!! \n\n";

}

}while(ch!= 7);

return 0;

}

cin>>str1;

cout<<"\nEnter the second string : ";

cin>>str2;

cout<<"\nFirst string is "<<str1;

cout<<"\nSecond string is "<<str2;

len = strcmp(str1,str2);

if(len == 0)

cout<<"\nBoth strings are equal";

else

{

if( len > 0)

cout<<"\nFirst string is greater than second string";

else

cout<<"\nSecond string is greater than first string";

}

break;

case 6 : cout<<"\nEnter the Main string : ";

cin>>str1;

cout<<"\nEnter the string to be checked-substring : ";

cin>>str3;

cout<<"\nMain string is "<<str1;

cout<<"\nSecond string is "<<str3;

//len = strstr(S,D);

if(len == 0)

cout<<"\nSecond is not a substring of main string";

else

cout<<"\nSecond string is a substring of first string with "<<len<<" occurences";

break;

case 7 : cout<<"\nGoodbye\n";

break;

default:cout<<"\nInvalid choice !! Try again!!! \n\n";

}

}while(ch!= 7);

return 0;

}

**Output:**

/\*

Enter the source string : hello

Source string is hello

1: String length

2: String Copy

3: String Concat

4: String Reverse

5: String compare

6: Substring

7: Exit

Enter ur choice : 1

Source string is hello

Length of the string is 5

1: String length

2: String Copy

3: String Concat

4: String Reverse

5: String compare

6: Substring

7: Exit

Enter ur choice : 2

Source string is hello

Copied string is hello

1: String length

2: String Copy

3: String Concat

4: String Reverse

5: String compare

6: Substring

7: Exit

Enter ur choice : 3

Enter the destination string : world

Source string is hello

Destination string is world

Concatinated string is worldhello

1: String length

2: String Copy

3: String Concat

4: String Reverse

5: String compare

6: Substring

7: Exit

Enter ur choice : 4

Source string is hello

Reverse string is olleh

1: String length

2: String Copy

3: String Concat

4: String Reverse

5: String compare

6: Substring

7: Exit

Enter ur choice : 5

Enter the first string : hello

Enter the second string : hello

First string is hello

Second string is hello

Both strings are equal

1: String length

2: String Copy

3: String Concat

4: String Reverse

5: String compare

6: Substring

7: Exit

Enter ur choice : 6

Enter the Main string : hello

Enter the string to be checked-substring : hel

Main string is hello

Second string is hel

Second string is a substring of first string with 1 occurences

1: String length

2: String Copy

3: String Concat

4: String Reverse

5: String compare

6: Substring

7: Exit

Enter ur choice : 7

Goodbye

\*/

.

|  |  |
| --- | --- |
| **Group** | **A** |
| **Assignment No** | **05** |
| **Title** | Write a C++ program to realize polynomial equation and perform operations. Write function  a) To input and output polynomials represented as b m x em + b m-1 x em-1 +..... +b 0 x e0 .  Your functions should overload the << and >> operators.  1. b) Evaluates a polynomial at given value of x  2. c) Add two polynomials  3. d) Multiplies two polynomials |
| **Date of completion** | 20/07/2017 |
| **Marks out of 10** | 9 |
| **Signature of staff** |  |

**Assignment No.A-5**

**Name: Ghodekar Prachi Shrikant.(Roll no. 12)**

**Objectives:**

* To explore and apply algebraic skills in manipulating polynomials.
* To implement addition, subtraction and multiplication of polynomials.

**Problem Statement:**

Write a C++ program to realize polynomial equation and perform operations. Write function

1. To input and output polynomials represented as as bmxem+ bm-1xem-1 +….. +b0xe0.

Your functions should overload the << and >> operators.

1. Evaluates a polynomial at given value of x
2. Add two polynomials
3. Multiplies two polynomials

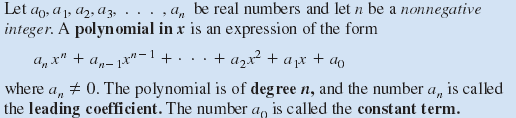
**Problem Definition:**

**Coefficient :** It is a constant that is either alone or being multiplied by an expression. Eg. 3 & -7 are coefficient for the term 3x5 and -7x2y.

**Exponent :** It is the power to which a number or expression is being raised. Eg. 2 & 3 are exponents for the term 62 and (-2c)3

Polynomial : A polynomial is an expression composed of coefficients and variables under addition, subtraction and multiplication and exponents on those variables must be non-negative integers.

In mathematics, a **polynomial** is an expression consisting of variables and coefficients which only employs the operations of addition, subtraction, multiplication, and non-negative integer exponents. An example of a polynomial of a single variable *x* is *x*2 − 4*x* + 7. An example in three variables is *x*3 + 2*xyz*2 − *yz* + 1.



Polynomials appear in a wide variety of areas of mathematics and science. For example, they are used to form polynomial equations, which encode a wide range of problems, from elementary word problems to complicated problems in the sciences; they are used to define **polynomial functions**, which appear in settings ranging from basic chemistry and physics to economics and social science; they are used in calculus and numerical analysis to approximate other functions. In advanced mathematics, polynomials are used to construct polynomial rings and algebraic varieties, central concepts in algebra and algebraic geometry.

**Evaluating a polynomial**: For the given value of *x*, figure out what value you get.

* **Evaluate 2*x*3 – *x*2 – 4*x* + 2 at *x* = –3**

2(–3)3 – (–3)2 – 4(–3) + 2   
  =  2(–27) – (9) + 12 + 2   
  =  –54 – 9 + 14   
  =  –63 + 14   
  =  **–49**

**Adding Polynomials :** To add two polynomials, simply combine like terms.

Egs : (5x2 + 6x – 3) + (2x2 – 7x – 9) = 7x2 – x – 12

**Multiplying Polynomials :** To multiply two polynomials multiply **each term** in one polynomial by **each term** in the other polynomial, add those answers together, and simplify if needed.

Egs : (x + 2) (x2 +4x – 7) = x3 + 6x2 + x - 14

**Algorithms :**

Write algorithms for following modules:

1. Algorithm for evaluating an polynomial

2. Algorithm for addition of two polynomials

3. Algorithm for multiplication of two polynomials

**Sample Input Output**

**MENU**

1. Enter the first polynomial

2. Enter the second polynomial

3.Addition

4.Multiplication

5.Display

**Enter input:**

Enter the no. of terms in the first polynomial : 3

Enter the coefficient & exponent of 1 term : 6 4

Enter the coefficient & exponent of 2 term : 2 3

Enter the coefficient & exponent of 3 term : 4 1

Enter the no. of terms in the second polynomial : 3

Enter the coefficient & exponent of 1 term : 7 3

Enter the coefficient & exponent of 2 term : 3 2

Enter the coefficient & exponent of 3 term : 9 0

**Sample Output:**

First polynomial = 6X^4 + 2X^3 + 4X^1

Second polynomial = 7X^3 + 3X^2 + 9X^0

Addition Result polynomial = 6X^4 + 9X^3 + 3X^2 + 4X^1 + 9X^0

Multiplication Result polynomial = 42X^7 + 32X^6 + 6X^5 + 82X^4 + 30X^3 + 36X

Evaluation : Enter the value of X = 1

First polynomial = 12

Second polynomial = 19

**Conclusion:**

Hence, we have implemented Polynomial operations like addition, multiplication and evaluation using sequential organization and also the concept of operator overloading.

**Oral Questions:**

Q1. How to overload << operator? Display object using output operator.

Q2. How to overload >> operator? syntax.

Q3. Which operators can’t be overloaded with friend functions?

Q4. Explain the different ways to represent polynomials using sequential organization?

Q5. Define Polynomial.

Q6. How a polynomial having maximum three variables can be represented in computer

memory by using an array? Represent the following polynomial in computer memory using

the structure defined by you:

**5x3y2z + 3x2y3z2+ 6xyz3 + 10.**

Q7. Represent the following polynomials using arrays

1. x2 + 5xy + y2 + y – x

2. 5x2 + 10xy + y2 + 20

Wh program to implement polynomial operations for a two variable polynomials.

Q8. Write a C++ program to implement polynomial operations for a three variable polynomials.

Q9. Write a C++ program to implement division operations for a single variable polynomial

Q10. Why there is a need to overload operators?

Q11. What the different operations that we do on polynomials?

Q12. Give applications of polynomials?

**Marks Obtained...........................**

**Sign of Teacher.............................**

**Code:**

#include<iostream>

#include<stdlib.h>

#include<math.h>

using namespace std;

// A[2i] --> coef A[2i + 1] is the exp

class polynomial

{

private :

int A[30];

int n;

public:

polynomial();

friend istream & operator >>(istream & din,polynomial & P);

friend ostream & operator << (ostream & dout,polynomial P);

void evaluation();

friend polynomial operator +(polynomial P1,polynomial P2);

friend polynomial operator \*(polynomial P1,polynomial P2);

};

polynomial :: polynomial()

{

n = 0;

}

void polynomial :: evaluation()

{

int i,X;

int Sum =0;

cout<<"\nEnter the value of X : ";

cin>>X;

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

{

Sum = Sum + (A[2\*i] \* pow(X ,A[2\*i+1]) );

}

cout<<"\n\t"<<"F("<<X<<") = "<<Sum;

}

istream & operator >>(istream & din,polynomial & P)

{

int i;

cout<<"\nEnter the no. of terms in the polynomial : ";

din>>P.n;

for(i =0 ;i < P.n; i++)

{

cout<<"\nEnter the coef & exp field of term "<<i+1<<" : ";

din>>P.A[2\*i]>>P.A[2\*i+1];

}

cout<<"\nAccepted successfully\n";

return din;

}

ostream & operator << (ostream & dout,polynomial P)

{

int i;

cout<<"Polynomial F(X) = ";

for(i =0 ;i < P.n; i++)

{

if( i != P.n - 1)

dout<<P.A[2\*i]<<"X^"<<P.A[2\*i+1]<<" + ";

else

dout<<P.A[2\*i]<<"X^"<<P.A[2\*i+1];

}

return dout;

}

polynomial operator +(polynomial P1,polynomial P2)

{

int i,j,k;

polynomial P3;

i = j = k = 0;

while(i < P1.n && j < P2.n)

{

if(P1.A[2\*i+1] == P2.A[2\*j+1]) // expononet same

{

if(P1.A[2\*i] + P2.A[2\*j] != 0 ) // coef addition not zero

{

P3.A[2\*k] = P1.A[2\*i] + P2.A[2\*j];

P3.A[2\*k + 1] = P1.A[2\*i+1];

k++;

}

i++,j++;

}

else

{

if(P1.A[2\*i+1] > P2.A[2\*j+1]) // add the greater expononet

{

P3.A[2\*k] = P1.A[2\*i];

P3.A[2\*k + 1] = P1.A[2\*i+1];

i++,k++;

}

else

{

P3.A[2\*k] = P2.A[2\*j];

P3.A[2\*k + 1] = P2.A[2\*j+1];

j++,k++;

}

}

}

while(i < P1.n )

{

P3.A[2\*k] = P1.A[2\*i];

P3.A[2\*k + 1] = P1.A[2\*i+1];

i++,k++;

}

while(j < P2.n)

{

P3.A[2\*k] = P2.A[2\*j];

P3.A[2\*k + 1] = P2.A[2\*j+1];

j++,k++;

}

P3.n = k;

return P3;

}

polynomial operator \*(polynomial P1,polynomial P2)

{

int i,j,k;

polynomial T1,T2; // Intial T1 , T2 contains 0 terms

for(i=0; i < P1.n; i++)

{

for(j=0;j<P2.n;j++)

{

T2.A[2\*j] = P1.A[2\*i] \* P2.A[2\*j];

T2.A[2\*j+1] = P1.A[2\*i+1] + P2.A[2\*j+1];

}

T2.n = P2.n;

T1 = T1 + T2; // addition

}

return T1;

}

int main()

{

int ch;

polynomial P1,P2,P3;

system("clear");

do

{

cout<<"\n\t\t\t1: Accept polynomial";

cout<<"\n\t\t\t2: Display polynomial";

cout<<"\n\t\t\t3: Evaulate the polynomial";

cout<<"\n\t\t\t4: Addition of two polynomials";

cout<<"\n\t\t\t5: Multiplication of two polynomials";

cout<<"\n\t\t\t6: Exit";

cout<<"\n\nEnter ur choice : ";

cin>>ch;

switch(ch)

{

case 1 : cout<<"\nInput First polynomial ";

cin>>P1;

cout<<"\nInput Second polynomial ";

cin>>P2;

break;

case 2 : cout<<"\nFirst ";

cout<<P1;

cout<<"\nSecond ";

cout<<P2;

break;

case 3 : cout<<"\nFirst ";

cout<<P1;

P1.evaluation();

cout<<"\nSecond ";

cout<<P2;

P2.evaluation();

break;

case 4 : cout<<"\nFirst ";

cout<<P1;

cout<<"\nSecond ";

cout<<P2;

P3 = P1 + P2;

cout<<"\nResultant ";

cout<<P3;

break;

case 5 : cout<<"\nFirst ";

cout<<P1;

cout<<"\nSecond ";

cout<<P2;

P3 = P1 \* P2;

cout<<"\nResultant ";

cout<<P3;

break;

case 6 : cout<<"\nGoodbye\n";

break;

default:cout<<"\nInvalid choice !! Try again!!! \n\n";

}

}while(ch!= 6);

return 0;

}

**Output**:

**/\***

1: Accept polynomial

2: Display polynomial

3: Evaulate the polynomial

4: Addition of two polynomials

5: Multiplication of two polynomials

6: Exit

Enter ur choice : 1

Input First polynomial

Enter the no. of terms in the polynomial : 3

Enter the coef & exp field of term 1 : 3

2

Enter the coef & exp field of term 2 : 2

1

Enter the coef & exp field of term 3 : 1

4

Accepted successfully

Input Second polynomial

Enter the no. of terms in the polynomial : 3

Enter the coef & exp field of term 1 : 3

2

Enter the coef & exp field of term 2 : 2

5

Enter the coef & exp field of term 3 : 1

4

Accepted successfully

1: Accept polynomial

2: Display polynomial

3: Evaulate the polynomial

4: Addition of two polynomials

5: Multiplication of two polynomials

6: Exit

Enter ur choice : 2

First Polynomial F(X) = 3X^2 + 2X^1 + 1X^4

Second Polynomial F(X) = 3X^2 + 2X^5 + 1X^4

1: Accept polynomial

2: Display polynomial

3: Evaulate the polynomial

4: Addition of two polynomials

5: Multiplication of two polynomials

6: Exit

Enter ur choice : 3

First Polynomial F(X) = 3X^2 + 2X^1 + 1X^4

Enter the value of X : 4

F(4) = 312

Second Polynomial F(X) = 3X^2 + 2X^5 + 1X^4

Enter the value of X : 5

F(5) = 6950

1: Accept polynomial

2: Display polynomial

3: Evaulate the polynomial

4: Addition of two polynomials

5: Multiplication of two polynomials

6: Exit

Enter ur choice : 6

Goodbye

\*/