CSE 1133 – Introduction to Programming

2018 – 2019 Fall Semester Project

Author:Cahid Alır

Number:180315006

1. m-th degree Summation

Write a program to calculate m-th degree summation with the integers m and n,

which are get from the user.

1.Recursive

2.Iterative

2. Fully sentence

(A fully sentence is a sentence using every letter of a given alphabet at least once.)

a. Write a program that checks the given sentence is fully or not.

b. Find the frequency of the letter that most occurred in the sentence.

3. Matrix Operations

Write a C program for matrix operations. You should display a simple console menu

to ask to user which operations will be done.

1) Transpose

2) Addition

3) Multiplication

1. m-th DEGREE SUMMATION

```c
int main()
{
    int endPoint, repeatCount;

    printf("Enter the end value:\n");
    scanf("%d",&endPoint);

    printf("Enter the repeat count:\n");
    scanf("%d",&repeatCount);

    while(endPoint<=0 || repeatCount<=0)
    {
        printf("Invalid values!");

        printf("Enter the end value:\n");
        scanf("%d",&endPoint);

        printf("Enter the repeat count:\n");
        scanf("%d",&repeatCount);
    }
```

At the beginning of program, user enters until which number he/she wants to add on and how many times it must be repeat.

After that program checks if the numbers valid or not.

There is two way of solving this problem:

1.1. Recursive

1.2. Iterative

1.1.Recursive

To solve this problem recursively we need two functions; one of them will add numbers from one to given number and another to repeat the process as much as user wants.

To Repeat The Process

```
int recursiveRepeat(int endPoint, int repeatCount)
{
    if(repeatCount==1)
    {
        return recursiveAddition(endPoint);
    }
    endPoint=recursiveAddition(endPoint);

    return recursiveRepeat(endPoint, repeatCount-1);
}
```

We will use the function which is seen above.

It will get the summation from another recursive function and assign that sum as new and point.Then it will decrease repeat counter and repeat itself until repeat counter is one.

When its one it will take summation one last time and return the result.

To Add Numbers

```
int recursiveAddition(int endPoint)
{
    if(endPoint==1)
    {
        return 1;
    }
    return endPoint + recursiveAddition(endPoint-1);
}
```

This function is basically adds end point to what comes before it and on...

1.2 Iterative

To solve this problem iteratively we will use a well known mathematical formula: n.(n+1)/2

```
int iterativeRepeat(int endPoint, int repeatCount)
{
    int counter, newEndPoint;

    for(counter=0;counter<repeatCount;counter++)
    {
        endPoint=endPoint*(endPoint+1)/2;
    }
    return endPoint;
}
```

As you can see on the function, in the for loop function assigns the sum of numbers from 1 to given number as new end point and repeats it as much as user wants.

The result of the both functions will be this:

```
Enter the end value:
5
Enter the repeat count:
3
7260 is sum(5,3)(recursive)
7260 is sum(5,3)(iterative)
```

If functions didn't work correctly we wouldn't find this result so it's not necessary to show all the steps.

2. FULLY SENTENCE

at the beginning we will take the sentence as an array.

```
char givenSentence[100];
gets(givenSentence);


if(isFullySentence(strlwr(givenSentence)) == 1)
{
    printf("Given sentence is a fully sentence\n");
} else {
    printf("Given sentence is not a fully sentence\n");
}
```

After that we will pass this array to another function to check if it's pangram or not but before that we will turn the every element of sentences to lowercase with the "strlwr" function.

If the answer from that function is 1 function will tell to user the sentence which he/she gave is pangram otherwise it will tell opposite.

```
char alphabet[] = {'a','b','c','d','e',
                   'f','g','h','i','j',
                   'k','l','m','n','o',
                   'p','q','r','s','t',
                   'u','v','w','x','y','z'};
```

We will declare alphabet as an array to compare with the sentence.

A-)Check if its fully or not

```
int i = 0 ,j = 0;
    int append = 0;
    while(sentence[i] != '\0')
    {
        for(j=0; j<26; j++)
        {
            if(sentence[i] == alphabet[j]) {
                alphabet[j] = '-';
                append++;
            }
        }
        i++;
    }
```

With this loops function will take the first element of the sentence and compare it with alphabet when a match found it will increase append and eliminate letter that matched from alphabet array.

If there is 26 append

program will return 1 otherwise it will return 0.

B-)Finding the most common letter

After declaration of alphabet array we will use this loops:

```
int frequency[26]={0};
    int repeatCount=0;
    char mostCommonWord='a';

    for(int i=0;sentence[i]!='\0';i++)
    {
        for(int j=0;j<26;j++)
        {
            if(alphabet[j]==sentence[i])
            {
                ++frequency[j];
            }
        }
```

It will check every element same as the function before that but it will increase the value of the element of the frequency array which will be in the equivalent position of the alphabet which has been checked.

After that

```
for(int i=0;i<26;i++)
    {
        if(frequency[i]>repeatCount)
        {
            repeatCount=frequency[i];
            mostCommonWord=alphabet[i];
        }
    }
    return mostCommonWord;
```

With this well known for loop will find the biggest element of frequency and assign the equivalent position of the alphabet array as most common letter.

We will return the result to main:

```
printf("%c",mostCommon(givenSentence));
```

The result will be like this

```
Two driven jocks help fax my big quiz.
Given sentence is a fully sentence
i
```

3. Matrix Operations

This function will be able to do three operations:

1.Transpose

2.Addition

3.Multiplication

At the beginning

We will show a menu to user to choose the operation which he/she desires.

```
int showMenu()
{
    int choice;

    printf("What would you like to do?\n");
    printf("1.Transpose\n2.Addition\n3.Multiplication\n4.Exit\n");
    scanf("%d",&choice);
    if(choice<1 || choice>4)
    {
        printf("Invalid operation!Please enter a valid one:\n");
        printf("1.Transpose\n2.Addition\n3.Multiplication\n4.Exit\n");
        scanf("%d",choice);
    }
    return choice;
}
```

It will return the choice to main:

```
switch(choice)
   {
       case TRANSPOSE:
           transposedArray();
           break;

       case ADD:
           additionOperator();
           break;

       case MULT:
           multiplicationOperator();
           break;

       case EXIT:
           return 0;
   }
```

3.1.Transpose

After getting the dimensions from the user:

```
int row, column, i, j;

printf("Enter the number of rows and columns:\n");
scanf("%d%d",&row, &column);

int standardArray[row][column];
```

program will generate an array:

```
srand(time(NULL));

for(i=0;i<row;i++)
{
    for(j=0;j<column;j++)
    {
        standardArray[i][j]=1+rand()%100;
    }
}
```

Program will assign a random number to every element of array.

Program will take every element of the first array and assign it to reverse dimensions.

```
int transposedArray[column][row];

for(i=0;i<row;i++)
{
    for(j=0;j<column;j++)
    {
        transposedArray[j][i]=standardArray[i][j];
    }
}
```

As you can see dimensions of the transposed array are reversed.

Then program will print every element of the both arrays:

```
printf("\n----------Standard Version Of Array----------\n\n");

for(i=0;i<row;i++)
{
    for(j=0;j<column;j++)
        {
            printf("\t%d",standardArray[i][j]);
        }
        printf("\n");
}


printf("\n----------Transposed Version Of Array----------\n\n");
for(i=0;i<column;i++)
{
    for(j=0;j<row;j++)
    {
        printf("\t%d",transposedArray[i][j]);
    }
    printf("\n");
}
```

Result will be like this:

```
----------Standard Version Of Array----------

        13      36      32
        16      16      81

----------Transposed Version Of Array----------

        13      16
        36      16
        32      81
```

3.2.Addition

Same as transpose program will get dimensions from user but this time it will generate two different random arrays.

```
int firstArray[row][column];
int secondArray[row][column];
int sumOfArrays[row][column];
srand(time(NULL));

for(i=0;i<row;i++)
{
    for(j=0;j<column;j++)
    {
        firstArray[i][j]=1+rand()%100;
    }
}

for(i=0;i<row;i++)
{
    for(j=0;j<column;j++)
    {
        secondArray[i][j]=1+rand()%100;
    }
}
```

When it comes to addition

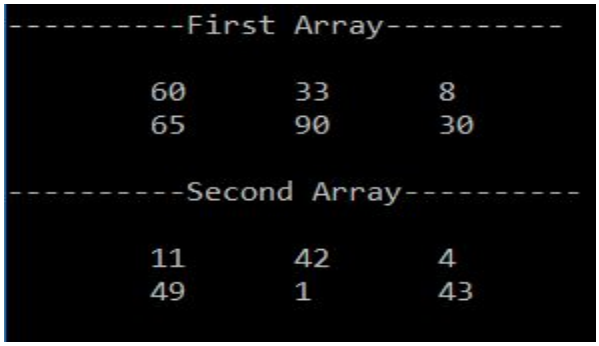
program will get elements from same positions of both arrays and assign the sum of them to a third arrays equivalent position.

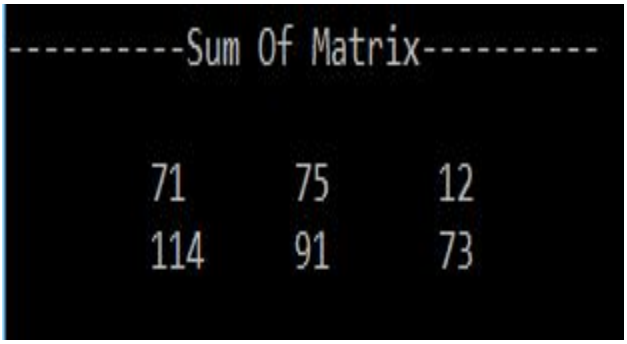
```
for(i=0;i<row;i++)
{
    for(j=0;j<column;j++)
    {
        sumOfArrays[i][j]=firstArray[i][j]+secondArray[i][j];
    }
}
```

After definitions of all three program will print them all

```
printf("\n----------First Array----------\n\n");

for(i=0;i<row;i++)
{
    for(j=0;j<column;j++)
        {
            printf("\t%d",firstArray[i][j]);
        }
        printf("\n");
}
printf("\n----------Second Array----------\n\n");

for(i=0;i<row;i++)
{
    for(j=0;j<column;j++)
        {
            printf("\t%d",secondArray[i][j]);
        }
        printf("\n");
}

printf("\n----------Sum Of Matrix----------\n\n");

for(i=0;i<row;i++)
{
    for(j=0;j<column;j++)
    {
        printf("\t%d",sumOfArrays[i][j]);
    }
    printf("\n");
}
```

Result will be like this:





3.3.Multiplication

This time program will get two different dimensions for two different arrays

```
int row1, column1, row2, column2, sum=0, i, j, k;

printf("Enter the number of rows and columns of first array:\n");
scanf("%d%d",&row1, &column1);

printf("Enter the number of rows and columns of first array:\n");
scanf("%d%d",&row2, &column2);

if(column1!=row2)
{
    printf("Invalid dimensions!\nPlease enter the same number to column of first array and row of second array");

    printf("Enter the number of rows and columns of first array:\n");
    scanf("%d%d",&row1, &column1);

    printf("Enter the number of rows and columns of first array:\n");
    scanf("%d%d",&row2, &column2);
}
```

Program check if the column of first array and the row of second array is same, if it's not asks for another dimensions because otherwise operation cannot be done.

Same as both operations before this one program generates two random arrays

```
int firstArray[row1][column1];
int secondArray[row2][column2];
int MultipliedArrays[row1][column2];
srand(time(NULL));

for(i=0;i<row1;i++)
{
    for(j=0;j<column1;j++)
    {
        firstArray[i][j]=1+rand()%100;
    }
}

for(i=0;i<row2;i++)
{
    for(j=0;j<column2;j++)
    {
        secondArray[i][j]=1+rand()%100;
    }
}
```

When it comes to multiplication program uses this operation:

```
for(i=0;i<row1;i++)
{
    for(j=0;j<column2;j++)
    {
        for(k=0;k<row2;k++)
        {
            sum=sum+firstArray[i][k]*secondArray[k][j];
        }
        MultipliedArrays[i][j]=sum;
        sum=0;
    }
}
```

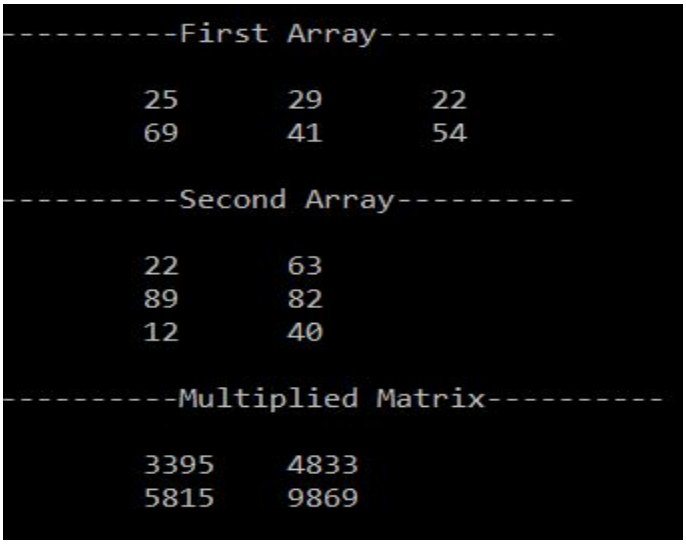
Program takes the first element of first row of first array, multiplies it with the first element of first row of second array and adds this value to sum and repeats this operation by increasing position and adding it to sum.

When the every element of row and column added assigns sum as an element of third element.

After that operation program prints all three array same as every array:

```
printf("\n----------First Array----------\n\n");

for(i=0;i<row1;i++)
{
    for(j=0;j<column1;j++)
        {
            printf("\t%d",firstArray[i][j]);
        }
        printf("\n");
}
printf("\n----------Second Array----------\n\n");

    for(i=0;i<row2;i++)
    {
        for(j=0;j<column2;j++)
            {
                printf("\t%d",secondArray[i][j]);
            }
            printf("\n");
    }

printf("\n----------Multiplied Matrix----------\n\n");

    for(i=0;i<row1;i++)
    {
        for(j=0;j<column2;j++)
        {
            printf("\t%d",MultipliedArrays[i][j]);
        }
        printf("\n");
    }
```

And result will be like this:



END OF THE PROJECT

Comments:

-At this project I learned how to efficiently use rand operators, arrays and function calling/passing.

-I haven't used pointers which is could be able more efficient if it's implemented but I couldn't be able to use it correctly.