Programming in C

Workshop India
Wilson Wingston Sharon
wingston.sharon@gmail.com

Hello World

- Simple program
- Traditional "first" program for many users

```
#include<stdio.h>

main()    // STRICT ANSI
{
          printf("Hello World\n");
          return 0;
}
```

```
#include<stdio.h>
#include<conio>
void main()  // allowed
{
          printf("Hello World\n");
          getch();
}
```

User input

```
#include<stdio.h>
main()
 {
        int number;
        printf("Enter a number\n");
        scanf("%d",&number);
        printf("Number entered by you is  %d\n", number);
        return 0;
}
```

- Gets some input from the user.
- Scanf function takes a **reference** to a type.
- **Exercise: take input of 9 numbers and display them**

Command line arguments

```
#include<stdio.h>

main(int argc, char *argv[])
{
  int c;

  printf("Number arguments passed: %d\n", argc);

  for ( c = 0 ; c < argc ; c++)
    printf("%d. argument passed is %s\n", c+1, argv[c]);

  return 0;
}
```

- Command line parameters are passed to the program from cmd command prompt

If/Else Comparision

```
#include<stdio.h>

main()
{
  int x = 1;
  if ( x == 1 )
    printf("x is equal to one.\n");
  else
    printf("Fail\n");
  return 0;
}
```

- Comparing variables for conditional operation.
- **== is comparision; = is assignment operator**
- Ex: **user enters a number. Program informs you if its even, odd or between 67 & 97.**

Array I/O

Declare an integer array

- size of array has to be a constant

Input how many elements the user requires in the array as (n)

- The user input should not be greater than the array size declared

Run a for loop from 0 to n and scanf it into &array[c]

Run another for loop and print the complete array

```
#include<stdio.h>
 main()
{
   int array[100], n, c;
   printf("Enter the number of elements in array\n");
   scanf("%d", &n);

   printf("Enter %d elements\n", n);

   for ( c = 0 ; c < n ; c++ )
      scanf("%d", &array[c]);

   printf("Array elements entered bu you are:\n");

   for ( c = 0 ; c < n ; c++ )
      printf("array[%d] = %d\n", c, array[c]);

   return 0;
}
```

Function

Decide whether you want the function body before or after main

- If before main function, write the function
- If after, write a function prototype before the main function

Decide on the return type and arguments and name them as well as the function appropriately

- Appropriate naming is important when sharing functions or using them long after the fact.

Write the function task and do a test call before using it in the program.

```
#include<stdio.h>

void my_function();

main()
{
  printf("Main function.\n");

  my_function();

  printf("Back in function main.\n");

  return 0;
}

void my_function()
{
  printf("Welcome to my function. Feel at home.\n");
}
```

Structures

Structure definitions are placed before the main function

- And don't forget the ; after the closing }

Declare an object of the defined structure

- struct <structure_name> <obj_name>;

Use the . (dot) operator to access structure members

- Variable.constant = 3;

```
#include<stdio.h>

struct programming
{
   float constant;
   char *pointer;
};
main()
{
  struct programming variable;
  char string[] = "Programming in Software Development.";
  variable.constant = 1.23;
  variable.pointer = string;

  printf("%f\n", variable.constant);
  printf("%s\n", variable.pointer);

  return 0;
}
```

Calculator

Declare all necessary variables.

- Making sure the variable used for result of division operator is float

Scanf the numbers

Perform all operations and display them

- Add
- Sub
- Mult
- div

```c
#include<stdio.h>

main()
{
  int first, second, add, subtract, multiply;
  float divide;
  printf("Enter two integers\n");
  scanf("%d%d",&first,&second);

  add = first + second;
  subtract = first - second;
  multiply = first * second;
  divide = first / (float)second;   //typecasting
  printf("Sum = %d\n",add);
  printf("Difference = %d\n",subtract);
  printf("Multiplication = %d\n",multiply);
  printf("Division = %.2f\n",divide);
  return 0;
}
```

Print current date

Include the <dos.h> header file

- This file has functions that read the current system clock date and time

Declare a struct date d;

- This structure is used to store the date information obtained

Use getdate(&d); to retrieve the system date

Print d.da_day as the current date variable

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>

main()
{
  struct date d;

  getdate(&d);

  printf("Current system date is %d/%d/%d",d.da_day,d.da_mon,d.da_year);
  getch();
  return 0;
}
```

Vowel checking

Declare a character (ch) and get an input using scanf() or getch()

- Scanf waits for enter
- Getch doesn't wait for enter

Check if the inputted character is an upper case or lowercase vowel from aeiou

- Else display not a vowel

```
#include<stdio.h>

main()
{
    char ch;

    printf("Enter a character\n");
    scanf("%c",&ch);

    if ( ch == 'a' || ch == 'A' || ch == 'e' || ch == 'E' || ch == 'i' || ch == 'I' || ch
=='o' || ch=='O' || ch == 'u' || ch == 'U')
       printf("%c is a vowel.\n", ch);
    else
       printf("%c is not a vowel.\n", ch);

    return 0;
}
```

Vowel checking

A better option?

- Try switch-case
- Convert input character to lowercase and then check only for lowercase
- Any other ideas?

Make it a function!

```
switch(ch)
  {
    case 'a':
    case 'A':
    case 'e':
    case 'E':
    case 'i':
    case 'I':
    case 'o':
    case 'O':
    case 'u':
    case 'U':
        printf("%c is a vowel.\n", ch);
        break;
    default:
        printf("%c is not a vowel.\n", ch);
  }
```

```
switch(ch)
  {
    case 'a':
    case 'A':
    case 'e':
    case 'E':
    case 'i':
    case 'I':
    case 'o':
    case 'O':
    case 'u':
    case 'U':
        printf("%c is a vowel.\n", ch);
        break;
    default:
        printf("%c is not a vowel.\n", ch);
  }
```

```
int check_vowel(char a)
{
   if ( a >= 'A' && a <= 'Z' )
      a = a + 'a' - 'A';   /* Converting to lower case */

   if ( a == 'a' || a == 'e' || a == 'i' || a == 'o' || a == 'u')
      return 1;

   return 0;
}
```

Leap year

What is a leap year?

- Divisible by 4, 100 AND 400

Input a year into an integer type

Use an if – else – if ladder to check for the given trye conditions

- Algorithm for checking follows
- http://en.wikipedia.org/wiki/Leap_year

Algorithm for leap year

Is year % 4 = 0?

- Yes – check next condition
- No – not leap year

Is year%100 = 0

- Yes – check next condition
- No – Leap year

Is year %400 = 0

- Yes – leap year
- No – not leap year

```
    int year;
    printf("Enter a year to check if it is a leap year\n");
    scanf("%d", &year);

if ( year%4 == 0)
{
        if(year%100 == 0)
        {
                if(year%400==0)
                        printf("Leap year!");
                else
                        printf("not Leap!");
        }
        else
        printf("Leap");
}
else
        printf("not leap year");
```

Fibonacci

Declare int variables : first second and next

- Also get number of terms to generate (n)

Loop from o to n incrementing by 1

if c<=1 don't add, just put next = c

Else: generate the next fibonacci term

- Next = first + second
- First = second
- Second = next

Print next and continue loop

```
printf("Enter the number of terms ");
  scanf("%d",&n);

  printf("First %d terms of fibonacci series are :-\n",n);

  for ( c = 0 ; c < n ; c++ )
  {
    if ( c <= 1 )
      next = c;
    else
    {
      next = first + second;
      first = second;
      second = next;
    }
    printf("%d\n",next);
  }
```

Sum of n numbers

Declare sum = 0 and n, var

- Scan n

Loop from 0 to n

Scan a number into var

- Var is a temporary variable used for the scanf function

Add sum and var and store it in var

Repeat loop till n

Print sum

```
#include<stdio.h>
#include<conio.h>

main()
{
  int n, sum = 0, c, var;
  printf("Enter the number of integers you want to add\n");
  scanf("%d",&n);
  printf("Enter %d numbers\n",n);

  for ( c = 1 ; c <= n ; c++ )
  {
    scanf("%d",&var);
    sum = sum + var;
  }
  printf("Sum of entered numbers = %d\n",sum);
  getch();
  return 0;
}
```

Reversing a number

Declare n and rev as int

- Scan n and initialise rev = 0

run a loop while n is not 0

rev = rev * 10

- Multiplication by 10 shifts the number left and puts a 0 in the ones place

rev = rev + n%10

- n%10 extracts the last digit of n and then we add it to rev

n = n/10

- Division by 10 and storing result in an integer types removes the ones digit of the number and shifts the number right

Loop ends and then you can print rev

```
#include<stdio.h>

main()
{
  int n, reverse = 0;

  printf("Enter a number to reverse\n");
  scanf("%d",&n);

  while( n != 0 )
  {
    reverse = reverse * 10;
    reverse = reverse + n%10;
    n = n/10;
  }

  printf("Reverse of entered number is = %d\n", reverse);

  return 0;
}
```

factorial

Input a number n and give it to the factorial function

- Make sure it is not 0 or negative

Make a factorial function

- Long return type and one int n argument

If n == 0

- Yes - return 1
- No – return n*factorial(n-1)

How this works

Say you call fact(5)

- n is 5 so if condition fails

Function should return 5*fact(4)

- Function is frozen in stack and fact(4) is called. Its result will be appended and then only fact(5) can return.

Fact(4) is called and should return 4*fact(3)

- Fact(5) is still waiting because fact(4) needs to call fact(3) before it can return

Fact(3) must return 3*fact(2) and Fact(2) must return 2*fact(1)

Fact(1) will return 1

- No more function calls so function can start returning

Fact(1) returns 1 to fact(2)

Fact(2) return 2*1 to fact(3)

Fact(3) returns 3*2*1 to fact(4)

Fact(4) returns 4*3*2*1 to fact(5)

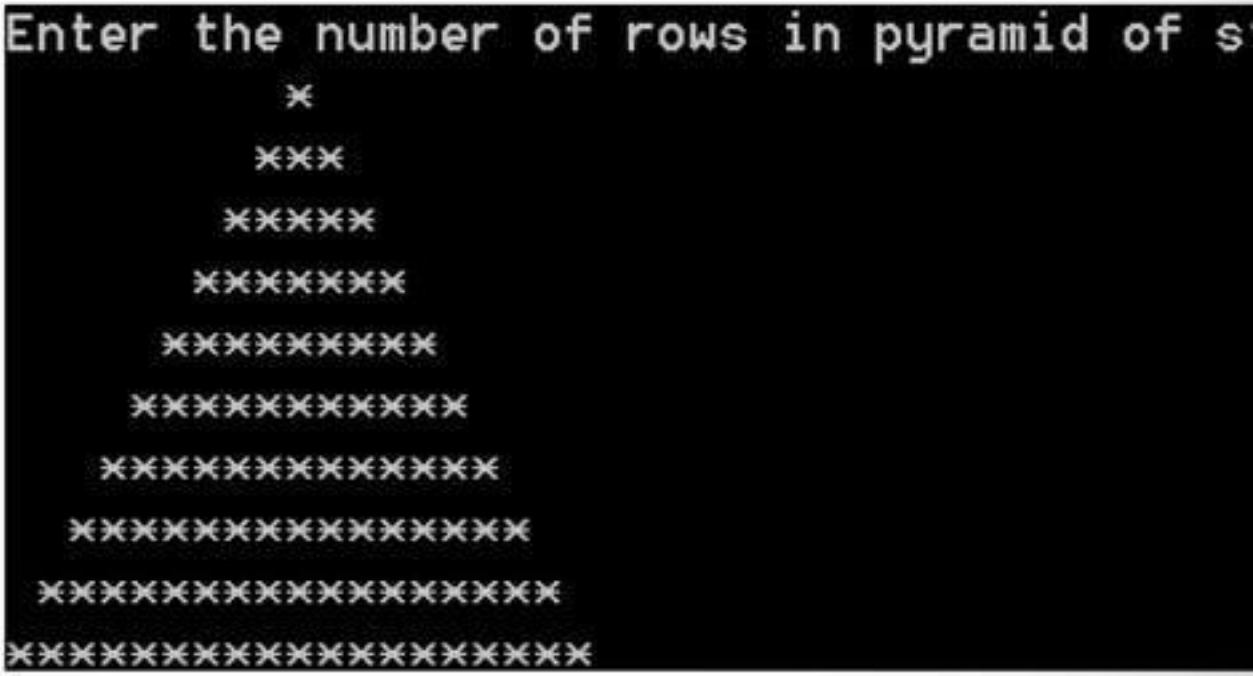
Fact(5) returns 5*4*3*2*1 to user

Recursive factorial

```
long factorial(int n)
{
  if(n==0)
    return(1);
  else
    return(n*factorial(n-1));
}
```

```
#include<stdio.h>
#include<conio.h>

main()
{
  int c, n, fact = 1;

  printf("Enter a number to calculate it's factorial\n");
  scanf("%d",&n);

  for( c = 1 ; c <= n ; c++ )
       fact = fact*c;

  printf("Factorial of %d = %d \n",n,fact);

  getch();
  return 0;
}
```

Pattern making a triangle



Lets say we want a 5 row triangle

Line 1 : 4 spaces and 1 star

Line 2: 3 spaces and 3 stars

Line 3: 2 spaces and 5 stars

Line 4: 1 spaces and 7 stars

Line 5: 0 space and 9 stars

Triangle pattern

First we take in as input number of rows we want in n

Make temp variable = n

Loop 1 : iterates over every line of the triangle

- Starts from row = 1 till row<=n

Inside, loop 1 first we print spaces for every line

- Print temp - 1 number of spaces " "
- Decrement temp

Then we print 2*row – 1 stars

Then a new line is printed

- row is incremented and loop1 starts again

```
#include<stdio.h>

main()
{
  int row, c, n, temp;
  printf("Enter the number of rows in pyramid of stars you wish to see ");
  scanf("%d",&n);
  temp = n;
  for ( row = 1 ; row <= n ; row++ )
  {
    for ( c = 1 ; c < temp ; c++ )
      printf(" ");
    temp--;
    for ( c = 1 ; c <= 2*row - 1 ; c++ )
      printf("*");
    printf("\n");
  }
  return 0;
}
```

Diamond pattern

Diamond pattern is like two triangles upside down

Same code for triangle is used except when drawing the second triangle we go backwards

- Spaces start from 1 and get incremented
- No of stars is 2*(n-rows)-1

Otherwise code is the same! Take a look!

```
#include<stdio.h>

main()
{
   int n, c, k, space = 1;

   printf("Enter number of rows\n");
   scanf("%d",&n);

   space = n - 1;
   for ( k = 1 ; k <= n ; k++ )
   {
      for ( c = 1 ; c <= space ; c++ )
         printf(" ");
      space--;
      for ( c = 1 ; c <= 2*k-1 ; c++)
         printf("*");
      printf("\n");
   }
```

```
space = 1;

   for ( k = 1 ; k <= n - 1 ; k++ )
   {
      for ( c = 1 ; c <= space; c++)
         printf(" ");

      space++;

      for ( c = 1 ; c <= 2*(n-k)-1 ; c++ )
         printf("*");

      printf("\n");
   }

   return 0;
}
```

Primality checking

- Problem statement: determine if a number is prime or not.

Input the number required to be checked for primality

- Give it to the checkprime() function

Loop from 2 to num-1

- Loop variable is c

If num %c =0

- Num is not prime return 0

After the loop if loop variable c =a

- Then the num is prime return 1

```
int check_prime(int a)
{
  int c;

  for ( c = 2 ; c <= a - 1 ; c++ )
  {
    if ( a%c == 0 )
          return 0;
  }
  if ( c == a )
    return 1;
}
```

Armstrong number

- A number is armstrong if the sum of cubes of individual digits of a number is equal to the number itself.

- For example 371 is an armstrong number as $3^3 + 7^3 + 1^3 = 371$.

- Some other armstrong numbers are: 0, 1, 153, 370, 407

armstrong number

Initialize num ,temp, remainder and sum=0

take a num from user

Store the num in temp

Take a loop till temp its not equal to 0

- Temp%10 store it in remainder
- Now add sum and (remainder to the power 3) and store it in sum
- Now temp/10 store it in temp

After the loop Check wether the num= sum

- If yes then its armstrong num
- Or else its not armstrong num

```
#include<stdio.h>
#include<conio.h>
main()
{
  int number, sum = 0, temp, remainder;
  printf("Enter a number\n");
  scanf("%d",&number);
  temp = number;
  while( temp != 0 )
  {
    remainder = temp%10;
    sum = sum + remainder*remainder*remainder;
    temp = temp/10;
  }
  if ( number == sum )
    printf("Entered number is an armstrong number.");
  else
    printf("Entered number is not an armstrong number.");
  getch();
}
```

Arrays – max and min

Declare the array,max,size,c and location=1

Take the num of elements from the user

First a loop must input all aray elements from the user

- The loop goes from 0 till array size

Initalize a variable called maximum to array[0]

Loop from 1 till array size to find maximum element

- Check with a condition each time whether the array is greater than maximum
- Store the array variable in maximum and increment the value of the size of the array each time

After the loop ends, maximum will hold the maximum element

```
#include<stdio.h>
main()
{
   int array[100], maximum, size, c, location = 1;
   printf("Enter the number of elements in array\n");  scanf("%d",&size);
   printf("Enter %d integers\n", size);
   for ( c = 0 ; c < size ; c++ )
      scanf("%d", &array[c]);
   maximum = array[0];
   for ( c = 1 ; c < size ; c++ )
   {
      if ( array[c] > maximum )
      {
         maximum = array[c];
         location = c+1;
      }
   }
printf("Maximum element is present at location number %d and it's value is
%d.\n", location, maximum);     return 0;
}
```

Binary search

Declare as variables first,last,middle ,n,search and array

- Scan array from user and sort it!

Scan variable to be searched and store it in search variable

Initialize the index variables first,last and middle

- Middle = (first + last)/2

start a while loop until first <= last

Check if search is on right side or left side of middle (x > a[mid] or x<a[mid])

- If x > a[mid] then min = mid+1
- If x<a[mid] then max = mid -1
- If x = a[mid] then we have found element and we stop loop

Then update mid itself and continue loop

If loop completes and first>last, then we have not found element

```
#include<stdio.h>

main()
{
  int c, first, last, middle, n, search, array[100];

  printf("Enter number of elements\n");
  scanf("%d",&n);

  printf("Enter %d integers\n", n);

  for ( c = 0 ; c < n ; c++ )
    scanf("%d",&array[c]);

  printf("Enter value to find\n");
  scanf("%d",&search);
  first = 0;
  last = n - 1;
  middle = (first+last)/2;
```

```
while( first <= last )
  {
    if ( array[middle] < search )
      first = middle + 1;
    else if ( array[middle] == search )
    {
      printf("%d found at location %d.\n", search, middle+1);
      break;
    }
    else
      last = middle - 1;

    middle = (first + last)/2;
  }
  if ( first > last )
    printf("Not found! %d is not present in the list.\n", search);

  return 0;
}
```

Reverse an array

Declare the variables n,c,d,temp and 2 arrays with same size

- Take the number of elements from the user (n)
- Take the array elements from the user

We start a single loop that goes forward through one array and backward through the other

- C = n+1 to 0 decrementing by 1 used for array a
- D = 0 to n-1 incrementing by 1used for array b

b[d] = a[c] copies a in reverse to b

We can copy b to a again so our original array a is reversed

```
#include<stdio.h>

main()
{
  int n, c, d, temp, a[100], b[100];
  printf("Enter the number of elements in array\n");
  scanf("%d",&n);
  printf("Enter the array elements\n");
  for ( c = 0 ; c < n ; c++ )
    scanf("%d",&a[c]);
  for ( c = n - 1, d = 0 ; c >= 0 ; c--, d++ )
    b[d] = a[c];
  for ( c = 0 ; c < n ; c++ )
    a[c] = b[c];
  printf("Reverse array is\n");
  for( c = 0 ; c < n ; c++ )
    printf("%d\n", a[c]);
  return 0;
}
```

Insert elements in array

Declare the variables array,position,c,n,value

- Take the number of elements from the user
- Take the values of the array

Take the position from user where the element to be added and store it in position

- Enter the value of the element to be added

Take a loop starting from n-1 to (position -1) and decrement each the value

- Increase the size of the array by 1 as we add a new element
- Now place the value to be added in that position.

The resultant array can be displayed

```
#include<stdio.h>

main()
{
    int array[100], position, c, n, value;

    printf("Enter number of elements in array\n");
    scanf("%d", &n);

    printf("Enter %d elements\n", n);

    for ( c = 0 ; c < n ; c++ )
       scanf("%d", &array[c]);

    printf("Enter the location where you wish to insert an element\n");
    scanf("%d", &position);

    printf("Enter the value to insert\n");
    scanf("%d", &value);
```

```
for ( c = n - 1 ; c >= position - 1 ; c-- )
        array[c+1] = array[c];

    array[position-1] = value;

    printf("Resultant array is\n");

    for( c = 0 ; c <= n ; c++ )
        printf("%d\n", array[c]);

    return 0;
}
```

```
#include<stdio.h>

main()
{
    int array[100], position, c, n;

    printf("Enter number of elements in array\n");
    scanf("%d", &n);

    printf("Enter %d elements\n", n);

    for ( c = 0 ; c < n ; c++ )
       scanf("%d", &array[c]);

    printf("Enter the location where you wish to delete element\n");
    scanf("%d", &position);
```

```
if ( position >= n+1 )
        printf("Deletion not possible.\n");
    else
    {
        for ( c = position - 1 ; c < n - 1 ; c++ )
            array[c] = array[c+1];

        printf("Resultant array is\n");

        for( c = 0 ; c < n - 1 ; c++ )
            printf("%d\n", array[c]);
    }

    return 0;
}
```

Matrix Addition

Declare all matrixes and matrix sizes, first, second and sum

- Matrix are 2D arrays and are declared as first[10][10]

A nested for loop from 0 to m and 0 to n is used to input matrix 1 and 2

- m and n are the sizes of the matrix

A nested for loop is used for matrix addition

- Loop 1 variable is c, goes from 0 to m
- Loop 2 variable is d, goes from 0 to n

sum[c][d] = first[c][d] + second[c][d]

- Print the sum matrix

```
#include<stdio.h>
#include<conio.h>

main()
{
  int m, n, c, d, first[10][10], second[10][10], sum[10][10];

  printf("Enter the number of rows and columns of matrix ");
  scanf("%d%d",&m,&n);
  printf("Enter the elements of first matrix\n");

  for ( c = 0 ; c < m ; c++ )
    for ( d = 0 ; d < n ; d++ )
      scanf("%d",&first[c][d]);

  printf("Enter the elements of second matrix\n");
```

```
for ( c = 0 ; c < m ; c++ )
    for ( d = 0 ; d < n ; d++ )
      sum[c][d] = first[c][d]+ second[c][d];

  printf("Sum of entered matrices:-\n");

  for ( c = 0 ; c < m ; c++ )
  {
    for ( d = 0 ; d < n ; d++ )
      printf("%d\t",sum[c][d]);

    printf("\n");
  }

  getch();
  return 0;
}
```

Matrix Transpose

Declare 2 matrixes and sizes

- A nested for loop from 0 to m and 0 to n is used to input matrix
- m and n are the sizes of the matrix

A nested for loop is used to transpose

- Loop 1 variable is c, goes from 0 to m
- Loop 2 variable is d, goes from 0 to n

Transpose[c][d] = matrix[d][c]

Print the transposed matrix

```
#include<stdio.h>

main()
{
  int m, n, c, d, matrix[10][10], transpose[10][10];

  printf("Enter the number of rows and columns of matrix ");
  scanf("%d%d",&m,&n);
  printf("Enter the elements of matrix \n");

  for( c = 0 ; c < m ; c++ )
  {
    for( d = 0 ; d < n ; d++ )
    {
      scanf("%d",&matrix[c][d]);
    }
  }
```

```
for( c = 0 ; c < m ; c++ )
  {
    for( d = 0 ; d < n ; d++ )
    {
      transpose[d][c] = matrix[c][d];
    }
  }
  printf("Transpose of entered matrix :-\n");
  for( c = 0 ; c < n ; c++ )
  {
    for( d = 0 ; d < m ; d++ )
    {
      printf("%d\t",transpose[c][d]);
    }
    printf("\n");
  }
  return 0;
}
```

Matrix multiplication

Declare all matrixes and matrix sizes, first, second and mul

- Input sizes and the two input matrixes

Check sizes of matrix for conformity

- m,n are sizes of matrix 1
- p,q are sizes of matrix 2
- Then if n == p only multiplication can take place

3 nested for loops are used

- Loop1 c from 0 to m
- Loop2 d from o to n
- Loop3 k from 0 to p

In loop3 sum = sum+first[c][k]*second[k][d]

After loop3 ends, in loop2

- mul[c][d] = sum
- Sum = 0

After all loops terminate, mul can be printed as resultant matrix

```
#include<stdio.h>
#include<conio.h>

main()
{
  int m, n, p, q, c, d, k, sum = 0;
  int first[10][10], second[10][10], mul[10][10];

  printf("Enter the number of rows and columns of first matrix\n");
  scanf("%d%d",&m,&n);
  printf("Enter the elements of first matrix\n");

  for (  c = 0 ; c < m ; c++ )
    for ( d = 0 ; d < n ; d++ )
      scanf("%d",&first[c][d]);

  printf("Enter the number of rows and columns of second matrix\n");
  scanf("%d%d",&p,&q);
```

```
if ( n != p )
    printf("Matrices with entered orders can't be multiplied with each
other.\n");
  else {
    printf("Enter the elements of second matrix\n");
    for ( c = 0 ; c < p ; c++ )
      for ( d = 0 ; d < q ; d++ )
        scanf("%d",&second[c][d]);

    for ( c = 0 ; c < m ; c++ )  {
      for ( d = 0 ; d < n ; d++ ) {
        for ( k = 0 ; k < p ; k++ ) {
          sum = sum + first[c][k]*second[k][d];
        }
        mul[c][d] = sum;
        sum = 0;
      }
    }
```

String I/O

Declare a character array to store the string

- Take the string from the user and scan it by %s

Now enter the value of the string as %s with the array name

```
#include<stdio.h>

main()
{
   char array[100];

   printf("Enter a string\n");
   scanf("%s",&array);

   printf("You entered the string %s\n",array);
   return 0;
}
```

Length of strings

Initialize the variables length and string name

Take the string from the user

- Use the function strlen for the string and store it in length

Now print the length of the string

```
#include<stdio.h>
#include<string.h>

main()
{
  char a[100];
  int length;

  printf("Enter a string to calculate it's length\n");
  gets(a);

  length = strlen(a);

  printf("Length of entered string is = %d\n",length);

  return 0;
}
```

Concatenation strings

Function takes two arguments as pointers

- These are character pointers that will point to the strings that are passed to the function

First we have to move the original pointer to the end of the first string

- When original is pointing to the end of the first string the while loop will fail. (last character is NULL)

Then we start another while loop that parses through the next string

- The while loop copies it character by character to the first string
- Both pointers are incremented

While loop will fail when the end of the add string is reached

- The string end NULL must be appended to the new string

After the function returns, the original string that was passed to the function would be changed

```
void concatenate_string(char *original, char *add)
{
  while(*original)
    original++;

  while(*add)
  {
    *original = *add;
    add++;
    original++;
  }
  *original = '\0';
}
```

Palindrome check

Initialize 2 strings a and b

- Get the string to be checked whether its palindrome or not

Use strcpy to copy the string to b

Now use strrev for b to reverse the string a

Now check with a loop using strcmp bt sting a and b if the resultant is 0

- Then it's a palindrome
- Or else its not a palindrome

```
#include<stdio.h>
#include<string.h>

main()
{
  char a[100], b[100];

  printf("Enter the string to check if it is a palindrome\n");
  gets(a);

  strcpy(b,a);
  strrev(b);

  if( strcmp(a,b) == 0 )
    printf("Entered string is a palindrome.\n");
  else
    printf("Entered string is not a pailndrome.\n");

  return 0;
}
```

Read & write files

Create a file pointer

- FILE *fp

Open the file object

- Fp = fopen("filename",'mode');

Fgetc() will return a character from the file

EOF will signify end of file.

```
include<stdio.h>
#include<conio.h>
#include<stdlib.h>

main()
{
  char ch, fname[25];
  FILE *fp;
  printf("Enter the name of file you wish to see ");
  gets(fname);
  fp = fopen(fname,"r");
  printf("The contents of %s file are :- \n\n",fname);

  while( ( ch = fgetc(fp) ) != EOF)
    printf("%c",ch);

  getch();
  fclose(fp);
  return 0;
}
```