TATA CONSULTANCY SERVICES

TATA

INTRODUCTION To C PROGRAMMING

Experience certainty. IT Services
Business Solutions
Outsourcing

CONTENTS

- OVERVIEW
- IMPORTANCE OF READING AND DISPLAYING DATA
- BASIC DATA TYPES
- COMMON MISTAKES
- SIMPLE EXAMPLE

.

BASIC DATA TYPES

1) **int**
Format specifier: %d

2) **double**
Format specifier: %lf

3) **char**
Format specifier: %c or %s for string

4)**float**
Format specifier: %f

IMPORTANCE

Sample Input:

A
23
45.678
Hema

Sample output 1:

A 23 45.678 Hema

Sample output 2:

A
23
45.678
Hema

Example:

Sample Input:

23
A
45.678
Hema

Sample output 1:
23 A 45.678 Hema

Sample output 2:
23
A
45.678
Hema

Decimal places

Sample Input:

23.78978
45.678
78.98

Sample output 1:(includes specific decimal places with and without rounding and rounding to nearest integer)
23.7897
45.68
79

Sample output 2:
23.789780
45.678000
78.980000

Type conversion

Implicit type

Explicit type

Assignments

- Write programs to scan different datatypes in different orders and displaying them in different orders other than the scanned order and in single line as well as different lines.
- Write a program to work on decimal values with different requirements as discussed.

TATA CONSULTANCY SERVICES

TATA

Happy Learning !

```
/*****************************************************************************

                    Online C Compiler.
            Code, Compile, Run and Debug C program online.
Write your code in this editor and press "Run" button to compile and execute it.

*****************************************************************************/

#include <stdio.h>

int main()
{
   int number=0;
   printf("Enter a number:");
   scanf("%d",&number);
   if(number%2==0){
      printf("%d is even number",number);
   }
   else
   {
      printf("%d is odd number",number);
   }
   return 0;
}

/*****************************************************************************
```

Welcome to GDB Online.

GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl, C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prolog. Code, Compile, Run and Debug online from anywhere in world.

```
*******************************************************************************/
#include <stdio.h>

int main()
{
   int n1, n2, n3;
   scanf("%d",&n1);
   scanf("%d",&n2);
   scanf("%d",&n3);
   if(n1>=n2 && n1>=n3)
   {
      printf("%d is greatest",n1);
   }
   else if(n2>=n1 && n2>=n3)
   {
      printf("%d is greatest",n2);
   }
   else if(n3>=n1 && n3>=n2)
   {
      printf("%d is greatest",n3);
   }
}
```

Alternative way of doing the programming using nested if

```
#include <stdio.h>

int main()
{
   int n1, n2, n3;
   scanf("%d",&n1);
   scanf("%d",&n2);
   scanf("%d",&n3);
   if(n1>n2)
   {
      if(n1>n3)
      {
          printf("%d is greatest",n1);
      }
      else
      {
          printf("%d is greatest",n3);
      }
   }
   else
   {
      if(n2>n3)
      {
          printf("%d is greatest",n2);
      }
      else
```

```
    {
        printf("%d is greatest",n3);
    }
  }
}


/*******************************************************************************

Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,
C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prolog.
Code, Compile, Run and Debug online from anywhere in world.

*******************************************************************************/

#include <stdio.h>
void main()
{
  int chk_year;

  printf("Input a year :");
  scanf("%d", &chk_year);
  if ((chk_year % 400) == 0)
    printf("%d is a leap year.\n", chk_year);
  else if ((chk_year % 100) == 0)
    printf("%d is a not leap year.\n", chk_year);
  else if ((chk_year % 4) == 0)
    printf("%d is a leap year.\n", chk_year);
```

```
  else
    printf("%d is not a leap year \n", chk_year);
}
```

Alternative way of doing the leap year program using logical and/or

```
#include<stdio.h>
void main() {
  int year;
  printf("Enter a year: ");
  scanf("%d", &year);
  if((year%4==0) && ((year%400==0) || (year%100!=0)))
  {
    printf("%d is a leap year", year);
  }
  else
  {
    printf("%d is not a leap year", year);
  }
}
```

/**********

Welcome to GDB Online.

GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl, C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prolog. Code, Compile, Run and Debug online from anywhere in world.

```
*****************************************************************************/
#include <stdio.h>

int main()
{
   char grade;
   scanf("%c", &grade);
   switch(grade)
   {
      case 'a':
      case 'A':
      printf("Excellent");
      break;



   }
}
```

Difference between & and &&

/***

Welcome to GDB Online.

GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl, C#, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prolog. Code, Compile, Run and Debug online from anywhere in world.

```
*****************************************************************************/
#include <stdio.h>

int main()
{
   int a=15;
   int b=20;
   int c= a&b;
   printf("%d",c);
}
```

Ternary Operator:

```
#include <stdio.h>

int main()
{
   int m = 5, n = 4;

   (m > n) ? printf("m is greater than n that is %d > %d",
                m, n)
         : printf("n is greater than m that is %d > %d",
                n, m);

   return 0;
}
```

Basic logical operator

```
#include <stdio.h>


main() {


    int a = 5;

    int b = 20;

    int c ;


    if ( a && b ) {

        printf("Line 1 - Condition is true\n" );

    }


    if ( a || b ) {

        printf("Line 2 - Condition is true\n" );

    }


    /* lets change the value of    a and b */

    a = 0;

    b = 10;


    if ( a && b ) {

        printf("Line 3 - Condition is true\n" );

    } else {

        printf("Line 3 - Condition is not true\n" );
```

```
    }

    if ( !(a && b) ) {
        printf("Line 4 - Condition is true\n" );
    }

}
```

Check greatest amon 2 numbers

```
#include <stdio.h>

void main()
{
    int num1, num2, num3;

    printf("Enter the values of num1, num2 and num3\n");
    scanf("%d %d %d", &num1, &num2, &num3);
    printf("num1 = %d\tnum2 = %d\tnum3 = %d\n", num1, num2, num3);
    if (num1 > num2)
    {
        if (num1 > num3)
        {
            printf("num1 is the greatest among three \n");
        }
        else
        {
```

```
            printf("num3 is the greatest among three \n");
        }
    }
    else if (num2 > num3)
        printf("num2 is the greatest among three \n");
    else
        printf("num3 is the greatest among three \n");
}
```

check greatest among 3 numbers

```
#include <stdio.h>


void main()
{
    int num1, num2, num3;


    printf("Enter the values of num1, num2 and num3\n");
    scanf("%d %d %d", &num1, &num2, &num3);
    printf("num1 = %d\tnum2 = %d\tnum3 = %d\n", num1, num2, num3);
    if (num1 > num2)
    {
        if (num1 > num3)
        {
            printf("num1 is the greatest among three \n");
        }
        else
```

```
        {
            printf("num3 is the greatest among three \n");
        }
    }
    else if (num2 > num3)
        printf("num2 is the greatest among three \n");
    else
        printf("num3 is the greatest among three \n");
}
```

C program to find grade

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int i;
    float mark, sum=0, avg;
    printf("Enter Marks obtained in 5 Subjects: ");
    for(i=0; i<5; i++)
    {
        scanf("%f", &mark);
        sum = sum+mark;
    }
    avg = sum/5;
    printf("\nGrade = ");
    if(avg>=91 && avg<=100)
```

```
        printf("A1");
    else if(avg>=81 && avg<91)
        printf("A2");
    else if(avg>=71 && avg<81)
        printf("B1");
    else if(avg>=61 && avg<71)
        printf("B2");
    else if(avg>=51 && avg<61)
        printf("C1");
    else if(avg>=41 && avg<51)
        printf("C2");
    else if(avg>=33 && avg<41)
        printf("D");
    else if(avg>=21 && avg<33)
        printf("E1");
    else if(avg>=0 && avg<21)
        printf("E2");
    else
        printf("Invalid!");
    getch();
    return 0;
}
```

reverse the number

```
#include <stdio.h>


int main()

{

	int a = 19876; //67891

	int c = a;

	printf("Before reverse :%d",a);

	int b = 0;

	while(a!=0){

		b = (b*10) + (a%10);

		a = a/10;

	}

	printf("\nAfter reverse:%d",b);

	if(c > b)

		printf("\n%d is greater",c);

	else

		printf("\n%d is greater",b);

	return 0;

}
```

sum of all digits

```
#include <stdio.h>


int main()

{
```

```
    int a = 19;
    int c = a;
    printf("Number is : %d",a);
    int b = 0;
    while(a!=0){
        b += (a%10);
        a = a/10;
    }
    printf("\nSum : %d",b);

    return 0;
}
```

average of all digits (rounded to specific decimal digits)

```
#include <stdio.h>

int main()
{
    int a = 19;
    printf("Number is : %d",a);
    int b = 0;
    int count = 0;
    while(a!=0){

        b += (a%10);
        a = a/10;
```

```
        count++;
    }
    float avg = b / count ;
    printf("\nAverage : %f",avg);

    return 0;
}
```

max/min digit in given number

```
#include <stdio.h>

int main()
{
    int a = 19756;
    printf("Number is : %d",a);
    int max = 0;
    int min = 99999999;
    while(a!=0){
        int b = a % 10;
        if(b > max)
            max = b;
        if(b < min)
            min = b;
        a = a/10;
    }
    printf("\nMin : %d and Max : %d",min,max);
```

```
    return 0;
}
```

Read n numbers and display maximum/minimum/average (only through iterations without array)

```
#include <stdio.h>


int main()
{
    int n = 0;
    scanf("%d",&n);
    int max = 0;
    int min = 99999999;
    int sum = 0;
    for (int i = 0 ; i < n ; i++){
        int a = 0 ;
        scanf("%d",&a);
        if(a > max)
            max = a;
        if(a < min)
            min = a;
        sum+=a;
    }
    float avg = sum/n;
    printf("\nSum : %d \nMax : %d \nMin: %d \nAvg: %f",sum,min,max,avg);
    return 0;
}
```

sum of odd/even digits

```
#include <stdio.h>


int main()

{

	int a = 19756;

	printf("Number is : %d",a);

	int even = 0;

	int odd = 0;

	while(a!=0){

		int b = a % 10;

		if(b%2 == 0)

			even += b;

		else

			odd += b;

		a = a/10;

	}

	printf("\nEven sum : %d and Odd sum : %d",even,odd);

	return 0;

}
```

Find max between numeric values before and after digits - E.g for input 4452.4567, max is 4567 and min is 4452

```
#include <stdio.h>

#include<conio.h>

int main()

{
```

```
    char str[50];
    gets(str);
    int val = 0;
    int a = 0 ;
    for (int i = 0 ; str[i] != '\0' ; i++){
        if(str[i] == '.'){
            val = a;
            a = 0;
        }
        else
            a = a*10 +    (str[i] - '0');

    }
    printf("Maximum value is : ");
    (val > a) ? printf("%d",val): printf("%d",a);
    return 0;
}
```

Finding the length of the string

```
#include <stdio.h>
int main()
{
char s[100];
int i;
printf("Enter a string: ");
scanf("%s", s);
for(i = 0; s[i] != '\0'; ++i);
printf("Length of string: %d", i);
return 0;
}
```

String Reverse without using library function or pointers

Alternative: Printing the sequence of character in reverse order

```
#include <stdio.h>
#include <string.h>
int main()
{
char str1[50], temp; // declare and initialize the size of string array.
int i = 0, j =0;
printf (" Enter a string to be reversed: ");
scanf( "%s", str1);  //GIRL
j = strlen (str1) - 1;  // j=3
for(i=j;i>=0;i--)
   {
      printf("%c",str1[i]);
   }
```

```
}
```

Reversing the string as a whole and printing the reversed string

```
#include <stdio.h>
#include <string.h>
int main()
{
char str1[50], temp; // declare and initialize the size of string array.
int i = 0, j =0;
printf (" Enter a string to be reversed: ");
scanf( "%s", str1);
j = strlen (str1) - 1;  // get the length of the string
// use while loop to define the condition
while ( i < j)
{
// use temp variable to store the characters of str1
temp = str1[j];
str1[j] = str1[i];
str1[i] = temp;
i++; // i incremented by 1
j--; // j is decremented by 1
}
printf (" The reversed of the string: %s", str1);
return 0;
}
```

Reading date as ddmmyyyy and displaying day, month and year

Simple version without loop:

```
#include <stdio.h>
#include <string.h>
#define MAX 1000
int main()
{
        char s[100];
        int i, j = 0;
   printf("\n Please Enter any String :  ");
        scanf("%s",s);
        printf("Day is : %c%c",s[0],s[1]);
   printf("\nMonth is : %c%c",s[2],s[3]);
   printf("\nYear is: %c%c%c%c",s[4],s[5],s[6],s[7]);
}
```

Loop is a must when extracting/playing with long list of characters.

```
#include <stdio.h>
#include <string.h>
int main()
{
int i=0;
char str1[50];
printf (" Enter a string");
scanf( "%s", str1);
```

```
printf("Day:");
for(i=0;i<2;i++)
{
   printf("%c",str1[i]);
}
printf("\nMonth:");
for(i=2;i<4;i++)
{
   printf("%c",str1[i]);
}
printf("\nYear:");
for(i=4;i<strlen(str1);i++)
{
   printf("%c",str1[i]);
}
return 0;
}
```

Count of vowels(Total Count)

```
#include <stdio.h>
int main()
{
  int c = 0, count = 0;
  char s[1000];
  printf("Input a string\n");
  scanf( "%s", s);
```

```
  while (s[c] != '\0') {
   if (s[c] == 'a' || s[c] == 'A' || s[c] == 'e' || s[c] == 'E' || s[c] == 'i' || s[c] == 'I' || s[c] =='o' || s[c]=='O' ||
s[c] == 'u' || s[c] == 'U')
     count++;
   c++;
  }
  printf("Number of vowels in the string: %d", count);
  return 0;
}
```

Count of vowels (for each vowel, however do not use arrays,run separate loops and display counts)

```
#include <stdio.h>
int main()
{
  int c = 0, counta = 0, counte = 0, counti = 0, counto = 0, countu = 0;
  char s[1000];
  printf("Input a string\n");
  scanf( "%s", s);
  while (s[c] != '\0') {
   if (s[c] == 'a' || s[c] == 'A')
     counta++;
   if (s[c] == 'e' || s[c] == 'E')
     counte++;
   if (s[c] == 'i' || s[c] == 'I')
     counti++;
   if (s[c] == 'o' || s[c] == 'O')
```

```
    counto++;
   if (s[c] == 'u' || s[c] == 'U')
    countu++;
   c++;
  }
  printf("\nNumber of a vowel in the string: %d", counta);
  printf("\nNumber of e vowel in the string: %d", counte);
  printf("\nNumber of i vowel in the string: %d", counti);
  printf("\nNumber of o vowel in the string: %d", counto);
  printf("\nNumber of u vowel in the string: %d", countu);
  return 0;
}
```

word count in sentence

```
#include <stdio.h>
#include <string.h>
#define MAX 1000
int main()
{
        char str[100];
        int i, totalwords, len;
        totalwords = 0;
        i = 0;
   printf("\n Please Enter any String :  ");
        fgets(str, MAX, stdin);
        while(str[i]!='\0')
{
```

```
if(str[i] == ' ')
{
totalwords++;
}
i++;
}
printf("\n The Total Number of Words in this String = %d",  totalwords+1); // incremented by 1 since total number of words will be total count of gap + 1
return 0;
}
```

Odd position substring

```
#include <stdio.h>
#include <string.h>
#define MAX 1000
int main()
{
        char str[100];
        int i, totalwords, len, j = 0;
        totalwords = 0;
        i = 0;
        char newStr[100];
   printf("\n Please Enter any String :  ");
        fgets(str, MAX, stdin);
        while(str[i]!='\0')
{
```

```
if(i%2==0)
{
newStr[j]=str[i];
j++;
}
i++;
}
printf("\n The odd position substring is = %s",  newStr);
return 0;
}
```

Simpler version without using if-else block

Start from index 0 and run upto the end. Increment the index by 2.

```
#include <stdio.h>
#include <string.h>
#define MAX 1000
int main()
{
        char str[100];
        int i, j = 0;
        char newStr[100];
   printf("\n Please Enter any String :  ");
        fgets(str, MAX, stdin);
        for(i=0; str[i]!='\0'; i=i+2)
        {
           printf("%c",str[i]);
        }
return 0;
```

```
}
```

For even position, simply start the loop from 1(index 1 means the 2nd character of the array i.e. the even position) and increment by 2 each time.

```
#include <stdio.h>

#include <string.h>

#define MAX 1000

int main()

{

        char str[100];

        int i, j = 0;

        char newStr[100];

   printf("\n Please Enter any String :  ");

        fgets(str, MAX, stdin);

        for(i=1; str[i]!='\0'; i=i+2)

        {

           printf("%c",str[i]);

        }

return 0;

}
```

Iterations

CONTENTS

- **While loop**
- **Do-while loop**
- **For-loop**

.

Example

Sample Input:

9

Sample output 1:
composite

Sample Input 2:
17

Sample output 2:
prime

While loop:

The while statement continually executes a block of statements while a particular condition is true.

.

Do-While loop:

The difference between do-while and while is that do-while evaluates its expression at the bottom of the loop instead of the top. Therefore, the statements within the do block are always executed at least once.

.

For loop:

For loops execute a sequence of statements multiple times

.

Example:

Write a program to print even numbers between given below range of numbers

Sample Input:

10
20

Sample output 1:
12
14
16
18

Assignments

- Try some requirements using all loops
- Make use of break and continue before and after print statements and check the output.

TATA CONSULTANCY SERVICES

TATA

Happy Learning !

Experience certainty. IT Services
Business Solutions
Outsourcing

TATA CONSULTANCY SERVICES

Basic Collections-Arrays

Experience certainty. IT Services
Business Solutions
Outsourcing

Write a program to print even numbers in given list of 5 integers(without arrays)

Sample Input:

10
13
15
18
20

Sample output 1:
10
18
20

Array: Collection of same data type

It can be declared as:

Datatype variable_name[size];

Examples:

Datatype a[10]={1,2,3,8,5,7};

Datatype b[5]={2,7,8,9,3};

Datatype c[]={values};

Datatype d[3];
d[0]=d[1]=d[2]=4;

Write a program to print sum of all integers of an array with size 5.

Input:

2
3
4
5
6

Output:

20

(Try with user input)

```
#include <stdio.h>
int main()
{
   int a[5]={2,3,4,5,6};
   int sum=0;
   for(int i=0;i<5;i++)
   {
      sum=sum+a[i];
   }
   printf("Sum = %d\n",sum);
   return 0;
}
```

Output: Sum = 20

Example

Write a program to print an array holding the sum of elements at respective indices of two same sized array.

Get elements from user for a[5] and b[5] and hold the result in c[5].

Let a[5]={1,2,1,2,1} and b[5]={2,1,2,1,2} Then, c[5]={3,3,3,3,3}

Print array elements of result in seperate lines with indices stating values of two arrays

Limitations :

- Fixed size
- Homogeneous
- Array doesn't check boundaries
- Shifting required for insertion or deletion

->Linked Lists

Assignments

- Try some requirements using all data types
- Count of integers greater than sum of elements in an array
- Count of integers greater than average of elements in an array
- Find maximum and minimum integers of an array
- Multiplication of all elements of an array
- Sum of even/odd digits of an array

TATA CONSULTANCY SERVICES

TATA

Happy Learning !

Experience certainty. IT Services
Business Solutions
Outsourcing

1) Find Maximum element in array

```
#include <stdio.h>
int main() {
  int n, max;
  int arr[100];
  printf("Enter the number of elements (1 to 100): ");
  scanf("%d", &n);

  for (int i = 0; i < n; ++i) {
    scanf("%d", &arr[i]);
  }
  max = arr[0];
  // storing the largest number to arr[0]
  for (int i = 1; i < n; ++i) {
    if (arr[i]>max) {
      max = arr[i];
    }
  }
  printf("Largest element = %d", max);
  return 0;
}
```

2) Find average of all numbers in array

```
#include <stdio.h>
int main() {
  int n;
```

```
 float sum = 0, avg;
 int arr[100];
 printf("Enter the number of elements (1 to 100): ");
 scanf("%d", &n);
 // taking input in array
 for (int i = 0; i < n; ++i) {
  scanf("%d", &arr[i]);
 }
 //adding all numbers to array
 for (int i = 0; i < n; ++i) {
     sum = sum + arr[i];
  }
 avg = sum / n ;
 printf("Average of numbers: %f", avg);
 return 0;
}
```

3) Create subset array of all odd numbers in array

Subarray:

A subarray is a contiguous part of array. An array that is inside another array. For example, consider the array [1, 2, 3, 4], There are 10 non-empty sub-arrays. The subarrays are (1), (2), (3), (4), (1,2), (2,3), (3,4), (1,2,3), (2,3,4) and (1,2,3,4)

```
#include <stdio.h>
int main() {
 int n, j=0;
```

```
 float sum = 0, avg;
 int arr[100];
 int oddArr[100];
 printf("Enter the number of elements (1 to 100): ");
 scanf("%d", &n);
 // taking input in array
 for (int i = 0; i < n; ++i) {
  scanf("%d", &arr[i]);
 }
 //Find the odd numbers in array
 for (int i = 0; i < n; ++i) {
     if(arr[i]%2==1)
     {
        oddArr[j] = arr[i];
        j++;
     }
  }
 for (int i = 0; i < j; ++i) {
  printf("\n%d", oddArr[i]);
 }
}
```

4) Swap List from middle (E.g. for list - 33 44 11 22 77 88 99 55, answer will be 77 88 99 55 33 44 11 22 as we swap first 4 and last 4. If list has odd item count then middle value remains at same position)

```
#include <stdio.h>
int main() {
 int n, z0, z1;
```

```
int arr[100], swapArr[100];
printf("Enter the number of elements (1 to 100): ");
scanf("%d", &n);
// taking input in array
for (int i = 0; i < n; ++i) {
 scanf("%d", &arr[i]);
}
for (int i = 0; i < n; ++i) {
   if(n%2==0)
   {
      z0=n/2;
      for(int i=z0, j=0; i<n, j<z0; i++, j++)
      {
         swapArr[i] = arr[j];
         swapArr[j] = arr[i];
      }
   }
   else
   {
      z0=((n-1)/2);
      swapArr[z0] = arr[z0];
      z1=(n+1)/2;
      for(int i=z1, j=0; i<n, j<z0; i++, j++)
      {
         swapArr[i] = arr[j];
         swapArr[j] = arr[i];
      }
   }
 }
```

```
  for (int i = 0; i < n; ++i) {
  printf("\n%d", swapArr[i]);
 }
}
```

5) Find count of prime numbers in the list

```
#include <stdio.h>
int main() {
 int n, checkPrime, countPrime = 0;
 int arr[100];
 printf("Enter the number of elements (1 to 100): ");
 scanf("%d", &n);
 // taking input in array
 for (int i = 0; i < n; ++i) {
  scanf("%d", &arr[i]);
 }
 //Find the count of prime numbers in array
 for (int i = 0; i < n; ++i) {
     checkPrime = arr[i];
     int flag = 0;
     if (checkPrime == 0 || checkPrime == 1)
         flag = 1;
     for(int j=2; j<checkPrime; j++)
```

```
    {
      if(checkPrime%j==0)
      {
        flag = 1;
        break;
      }
    }
    if(flag == 0)
    {
      countPrime++;
    }
  }
  printf("\n%d", countPrime);
}
```

Search input value in given List if numbers/strings using iterations

```
/*****************************************************************************
Search input value in given List if numbers/strings using iterations
*****************************************************************************/


#include <stdio.h>


int main()
{
    int arr[10] = {1,2,3,4,5,6,7,8,9,10};
    int num , i = 0;
    printf("Enter a number to be search : ");
    scanf("%d",&num);
    //For loop
    for (i = 0 ; i < 10 ; i++){
        if(arr[i] == num){
            printf("The number found at index = %d",i);
            break;
        }
    }
    if(i == 10){
        printf("%d not found in the list",num);
    }
    return 0;
}
```

Search numeric values from array between given range of two input values

```
/*****************************************************************************
Search numeric values from array between given range of two input values
*****************************************************************************/


#include <stdio.h>


int main()
{
	int arr[10] = {1,2,3,4,5,6,7,8,9,10};
	int start, end;
	printf("Enter the start index : ");
	scanf("%d",&start);
	printf("Enter the end index : ");
	scanf("%d",&end);
	//For loop
	for (int i = 0 ; i < 10 ; i++){
		if(arr[i]>start && arr[i]<end){
			printf("%d ",arr[i]);
		}
	}
	return 0;
}
```

Search how many numbers in given array are divisible by input value. Display those numbers

```
/*****************************************************************************
```

Search how many numbers in given array are divisible by input value. Display those numbers

**********/

```
#include <stdio.h>

int main()
{
    int arr[10] = {1,2,3,4,5,6,7,8,9,10};
    int num;
    printf("Enter the number : ");
    scanf("%d",&num);

    //For loop
    for (int i = 0 ; i < 10 ; i++){
        if(arr[i]%num == 0){
            printf("%d ",arr[i]);
        }
    }
    return 0;
}
```

Sort numeric array (use any basic sorting algorithm)

/**********

Sort numeric array (use any basic sorting algorithm)

**********/

```
#include <stdio.h>


int main()

{

	int arr[10] = {11,10,7,4,1,20,15};

	//For loop

	for (int i = 0 ; i < 7 ; i++){

		for (int j = 0 ; j < 6 ; j++){

			if(arr[j] > arr[j+1]){

				int c = arr[j];

				arr[j] = arr[j+1];

				arr[j+1] = c;			}

		}


	}


	for(int i = 0 ; i < 7 ; i++){

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

	}

	return 0;

}
```

Find second largest value in numeric array

```
/*****************************************************************************
Find second largest value in numeric array
*****************************************************************************/
```

```
#include <stdio.h>


int main()

{

	int arr[10] = {11,10,7,4,1,20,15};

	//For loop

	int max=0 , Second_max=0 ;

	for (int i = 0 ; i < 7 ; i++){

		if(i == 0){

			max = arr[i];

		}

		else if(max < arr[i]){

			max = arr[i];

		}

		else if(arr[i]>Second_max && arr[i]<max){

			Second_max = arr[i];

		}

		else{

			;

		}

	}

	printf("Second largest in the array is: %d",Second_max);

	return 0;

}
```

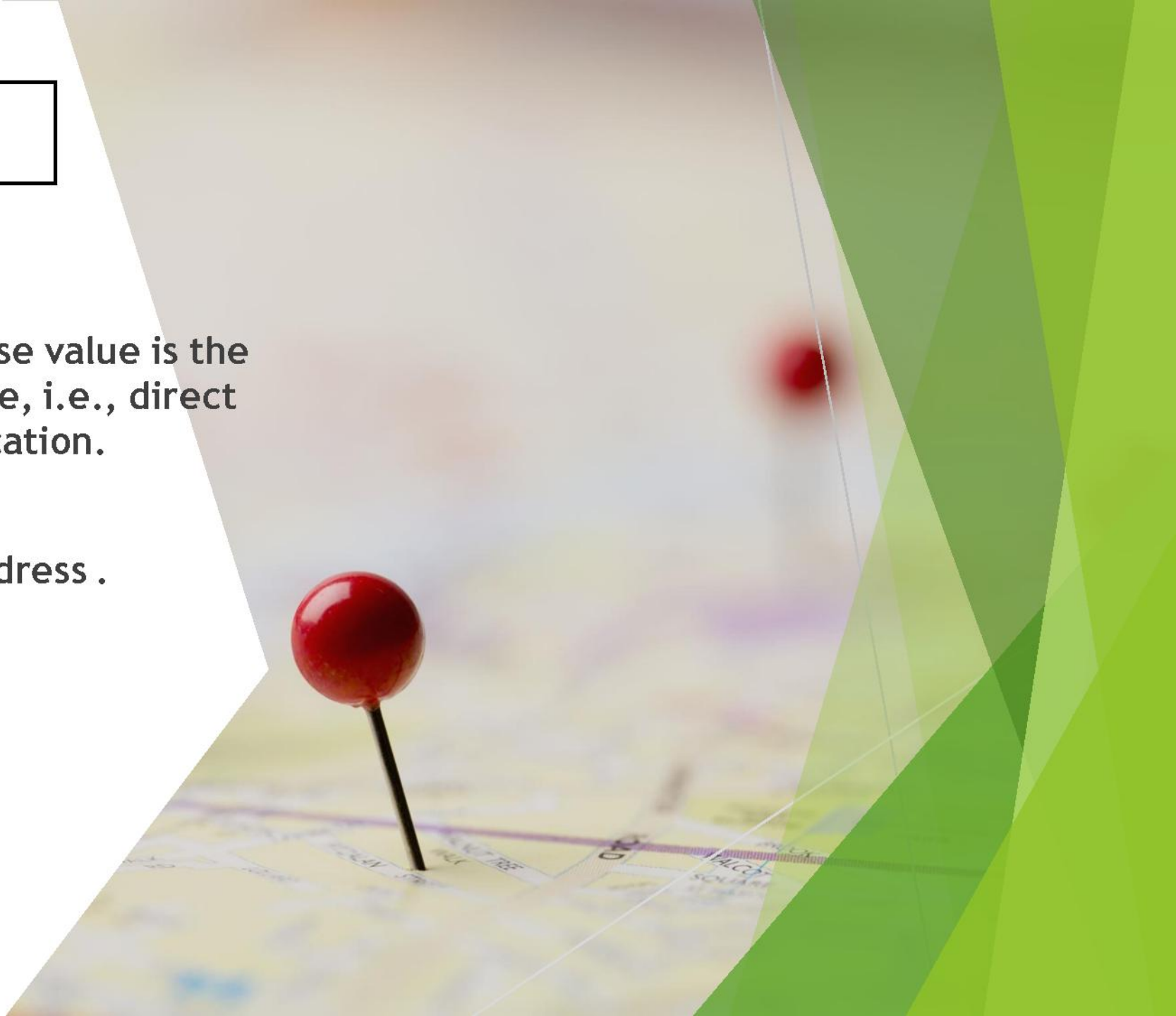
Linked Lists

Pre-requisites

- Pointers
- Structures

Pointers

- A pointer is a variable whose value is the address of another variable, i.e., direct address of the memory location.
- Datatype *ptr;
- Variable ptr holds the address .



Structures

- Structure in c is a user-defined data type that enables us to store the collection of different data types.
- Struct name{variables};

Linked lists

- A Linked List is a linear data structure.
- Every linked list has two parts, the data section and the address section that holds the address of the next element in the list, which is called a node.
- The size of the linked list is not fixed, and data items can be added at any locations in the list.
- The disadvantage is that to get to a node, we must traverse to all the way from the first node to the node that we require. The Linked List is like an array but unlike an array, it is not stored sequentially in the memory.

Single linked lists

For single linked list:

```
struct node {
    int data;
    struct node *next;

};
struct node *ptr;
Access data: ptr->data
Access next node(address field of current node)
 ptr->next
```

Double linked lists

- **For Double linked list :**
- ```
  struct Node {
      int data;
      struct Node *next;
      struct Node *prev;
  };
  ```
- struct Node *ptr;
- Acess data : ptr->data
- Acess next node : ptr->next
- Access previous node : ptr->prev
  ```

Assignments

- Try some requirements on both single and double linked lists like:
- Inserting at specific position
- Deleting at specific position
- Reversing elements

Happy learning!

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

  struct node
  {
   int data;
   struct node *next;
  };
 struct node *temp,*head,*newnode;
 void create();
 void insert();
 void delete_at_end();
 void reverse();

 struct dnode
 {
    int data;
    struct dnode *prevnode;
    struct dnode *nextnode;
 };
 struct dnode *dhead,*dnewnode,*dtemp;
 void create_doublylinkedlist();
int main()
{

create();
insert();
delete_at_end();
reverse();
create_doublylinkedlist();
   return 0;
}

 void create_doublylinkedlist()
 {
    int n;
  printf("----------------doubly linked list\n");
  scanf("%d",&n);

  for(int i=0;i<n;i++)
  {
     dnewnode=malloc(sizeof(struct dnode));
     scanf("%d",&dnewnode->data);
    printf("%d\t",dnewnode->data);
    if(i==0)
    {
       dhead=dtemp=dnewnode;
    }
```

```
    else{
        dtemp->nextnode=dnewnode;
        dnewnode->prevnode=dtemp;
        dtemp=dnewnode;
    }


  }
   while(dtemp!=0)
  {
     printf("\nreverse %d\t %d\t %d \n",dtemp->data,dtemp->nextnode,dtemp);
     dtemp=dtemp->prevnode;
  }


  dtemp=dhead;
   while(dtemp!=0)
  {
     printf("\nlinear %d\t %d\t %d \n",dtemp->data,dtemp->nextnode,dtemp);
     dtemp=dtemp->nextnode;
  }

 }
 void reverse()
 {   printf("----------------reverse");
    struct node *m,*prev,*cur,*nextnode;
    prev=0;
    cur=nextnode=head;
    while(nextnode!=0)
    {
        nextnode=nextnode->next;
        cur->next=prev;
        prev=cur;
        cur=nextnode;

    }

    m=prev;
     while(m!=0)
  {
     printf("\n%d %d %d \n",m->data,m->next,m);
     m=m->next;
  }

 }
void delete_at_end()
{
   printf("----------------delete at end");
```

```
  struct node *pre,*m;
   temp=head;
     while(temp->next!=0)
  {
     pre=temp;
     temp=temp->next;
  }
  pre->next=0;
  free(temp);
  m=head;
    while(m!=0)
  {
     printf("\n%d %d %d \n",m->data,m->next,m);
     m=m->next;
  }


}

void insert()
{
   printf("-------------adding");
   struct node *m;
   temp=head;
    while(temp->next!=0)
  {
          temp=temp->next;
  }
  // printf("%d \t %d\n",temp,temp->next);

   newnode=malloc(sizeof(struct node));
   scanf("%d",&newnode->data);
  // printf("%d\n",newnode->data);
   temp->next=newnode;
   //printf("%d \t %d\n",temp,temp->next);
   temp=newnode;
//printf("%d \t %d\n",temp,temp->next);
   m=head;
    while(m!=0)
  {
     printf("\n%d %d %d \n",m->data,m->next,m);
     m=m->next;
  }
}

void create()
{
   int n;
```

```
  //printf("Enter no. of nodes you want to create");
  scanf("%d",&n);

  for(int i=0;i<n;i++)
  {
     //printf("\nstruct node size :%d\n",malloc(sizeof(struct node)));
     //printf("\nstruct node size :%d\n",(struct node *)malloc(sizeof(struct node)));
    newnode=malloc(sizeof(struct node));
    //printf("%d\n new node next",newnode->next);
    scanf("%d",&newnode->data);
   printf("%d\t",newnode->data);
   if(i==0)
   {
      head=temp=newnode;
   }
   else{
      temp->next=newnode;
      temp=newnode;
   }


  }

  temp=head;
  //printf("\n head data %d\n",temp->data);
    printf("\ndata\t next\t address");
  while(temp!=0)
  {
    printf("\n%d\t %d\t %d \n",temp->data,temp->next,temp);
    temp=temp->next;
  }


}
```

In this session we discussed:

- Concept of class and objects and utility in the real world.
- Declaration of class and object.
- Using constructors in C++.
- Methods & method overloading within class.
- Considered examples of object comparison through attributes, like - we compared two objects based on age. We learnt passing class objects as method parameters, returning class objects by method and printing desired value.

1) Basic Program on class & objects

```
#include <bits/stdc++.h>

using namespace std;

class Geeks

{

   public:

    // Data Members

   string geekname;

    // Member Functions()

   void printname()

   {

     cout << "Geekname is: " << geekname;

   }

};


int main()

{

   Geeks obj1;

   obj1.geekname = "Abhi";

   obj1.printname();

   return 0;
```

```
}
```

2) Comparison of two objects based on ages.

```
#include <bits/stdc++.h>
using namespace std;
class Geeks
{
   public:
    // Data Members
   string geekname;
   int age;
    // Member Functions()
   void printname()
   {
     cout << "Younger Member is: " << geekname;
   }
   Geeks static returnYounger(Geeks obj1, Geeks obj2)
   {
      if(obj1.age<obj2.age)
      {
         return obj1;
      }
      else
      {
         return obj2;
      }
   }
};
```

```
int main()
{
   Geeks obj1;
   obj1.geekname = "Abhi";
   obj1.age = 30;
   Geeks obj2;
   obj2.geekname = "Priya";
   obj2.age = 32;
   Geeks youngerObj;
   youngerObj = Geeks::returnYounger(obj1, obj2);
   youngerObj.printname();
   return 0;
}
```

Given Assignment:

Take two student objects having parameters as name and marks. Take a method that accepts the defined objects and returns the student object with higher marks.

Inside the main method, print the student's name who scored higher marks.

3) Using Constructor:

```
 #include <iostream>
using namespace std;

class Point
{
private:
   int x, y;
```

```
public:
    // Parameterized Constructor
    Point(int x1, int y1)
    {
        x = x1;
        y = y1;
    }

    int getX()
    {
        return x;
    }
    int getY()
    {
        return y;
    }
};

int main()
{
    // Constructor called
    Point p1(10, 15);

    // Access values assigned by constructor
    cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();

    return 0;
}
```

Constructor Example 2: Defining more number of objects, each having its own copy of variables.

```
#include <iostream>
using namespace std;

class Point
{
private:
   int x, y;
public:
   // Parameterized Constructor
   Point(int x1, int y1)
   {
      x = x1;
      y = y1;
   }
   int getX()
   {
      return x;
   }
   int getY()
   {
      return y;
   }
};

int main()
{
```

```
   // Constructor called
   Point p1(10,15);
   Point p2(100,150);
   Point p3(1000,1500);
   // Access values assigned by constructor
   cout << "p2.x = " << p2.getX() << ", p3.y = " << p3.getY();
   return 0;
}
```

4)

Method Overloading - Example 1

```
#include <iostream>
using namespace std;

void print(int i) {
  cout << " Here is int " << i << endl;
}
void print(double  f) {
  cout << " Here is float " << f << endl;
}
void print(char const *c) {
  cout << " Here is char* " << c << endl;
}

int main() {
  print(10);
  print(10.10);
  print("ten");
```

```
  return 0;
}
```

Method Overloading - Example 2

```
#include <iostream>
using namespace std;

void area(int i) {
  cout << " Area of square is " << i*i << endl;
}
void area(double  f) {
  cout << " Area of circle " << 3.14*f*f << endl;
}
void area(int a, int b) {
  cout << "Area of rectangle is " << a*b << endl;
}
 int main() {
  area(10);
  area(10,20);
  area(10.2);
  return 0;
}
```