# Recursion

in recursion we try to solve a bigger problem by finding out sol of smaller sub problem . we represent these problem in a form of function and these function call itself to solve smaller sub problems.

## factorial using recursion

#include <iostream>

using namespace std;

int factorial(int n){

    if (n==1) // base

    return 1;

    int ans=n\*factorial(n-1); //assumption // self work

    return ans;

}

**Space**

stack frame

n stack frame and inside each frame n such variable

therefore O(n)

**Time Complexity**

no of time function called \* time consumed by one function

inside one function it is constant task running 1

and no of function is n times

T.C=n\*1

therefore, time com O(n)

## Fibonacci using recursion

0 1 1 2 3 5 8 13 21.......

nth term is the sum of n-1 and n-2

and we know 1st term is 0 and second term is 1

int fibonacii(int n){

    if (n==0 or n==1)

    return n;

    int ans = fibonacii(n-1) + fibonacii(n-2);

    return ans;

}

## sum of n natural numbers using recursion

int sum(int n){

    if (n<10)

    return n;

    int ans=(n%10)+sum(n/10);

    return ans;

}

## given two number p and q find the p^q

### method1

int expo(int p,int q){

    if(q==0)

    return 1;

    int ans=p\*expo(p,q-1);

    return ans;

}

Time complexity-O(q)

### Method2

int expo2(int p,int q){

    if(q==0)

    return 1;

    else if(q/2==0){

        int result = expo2(p,q/2);

        return result\*result;

    }

    else{

       int result= expo2(p,q-1/2);

       return p\*result\*result; //reuse

    }

}

# array in recursion

f(arr,index) arr = given array, index = current index we are pointing in array

print all the elements of the array recursively

base case- yah toh array empty hai yah index last element par pahuch gya

void arr(int array[],int size,int index){

    if(size<0 || index==size) return;

    cout<<array[index]<<" ";

    arr(array,size,index+1);

}

TC O(n)

SC O(n)

## print max of the array using recursion [3,10,3,2,5]

recursion kaisey ise hogi mai ek index par hu aur merko usey agey ki array ka max la do toh max aur index value ko conpare kar dungi

int maxi(int arr[],int size,int index){

    if (index==size-1) return arr[index];

    return max(arr[index],maxi(arr,size,index+1));

}

int f2(int arr[],int size,int index){

    if (index==size-1) return arr[index];

    int n=f2(arr,size,index+1);

    int ans=((arr[index]>n)?arr[index]:n);

    return ans;

}

## 6. sum of the array

int sum(int arr[],int size,int index){

    if (index==size-1) return arr[index];

    return (arr[index] + sum(arr,size,index+1));

}

## 7. check wheather the array is sorted or not

bool check(int arr[],int n){

    if(n==1) return true;

    return((arr[n-1]>arr[n-2]) && check(arr,n-1));

}

int main(){

    int arr[]={1,2,3,4,5};

    int size=sizeof(arr)/sizeof(arr[0]);

    cout<<check(arr,size);

}

# strings using recursion

## 8. remove all the occurance of "a" from the string s

string str(string &s,int index){

    if(s[index]=='\x0') return "\x0";

    string curr = "";

    curr+=s[index]; // bcoc s[index] is a character and we need to return a string

    return (((s[index] == 'a') ? "" : curr)+str(s,index+1));

}

## 9. move all the x at the end of the string

string moveX(string s){

    if(s.length()==0) return "";

    char ch = s[0];

    string ans = moveX(s.substr(1));

    if(ch=='x') return ans+ch;

    return ch+ans;

}

int main(){

    cout<<moveX("axxbdcefxhix");

}

## 10. replace "pi" with 3.14

string replace(string str){

    if(str.length()==0) return "";

    if(str[0]=='p'&& str[1]=='i'){

        return "3.14"+replace(str.substr(2));

    }

    else{

       string ros=string(1,str[0]);

       return ros+replace(str.substr(1));

    }

}

int main(){

    cout<<replace("pippxxppiixipi");

}

## 11. reverse a string

void reverse(string s,int n){

    if(n==(0)) return;

    cout<<s[n-1];

    reverse(s,n-1);

}

int main() {

    reverse("binod",5);

}

string reverse(string s){

    if(s.length()==0) return "";

    string ros=string(1,s[0]);

    return reverse(s.substr(1))+ros;

}

int main(){

    cout<<reverse("Hello World");

}

## 12. write a program to check whether is palindrome or not

bool palindrome(string s,int size,int index){

    if (size==1) return true;

    if(s[index]!=s[size-1]) return false;

    if(index<size) return palindrome(s,index+1,size-1);

}

bool isPalindrome(string s, int i){

    if(i > s.size()/2) return true ;

    return s[i] == s[s.size()-i-1] && isPalindrome(s, i+1) ;

}

## 13.remove all consecutive duplicates from the string

string duplicate(string s){

    if(s.length()==0) return "";

    char ch=s[0];

    string ans = duplicate(s.substr(1));

    if(ch==ans[0]) return ans;

    return ch+ans;

}

int main(){

    cout<<duplicate("aaaabbbbbbeeeccddd"); //abecd

    cout<<duplicate("aaaabbbbbbaaaeeeccddd"); //abaecd

}

## 14. generate all substring of a string

//

void subseq(string s,string ans){

    if(s.length()==0){

        cout<<ans<<endl;

        return;

    }

    char ch=s[0];

    string ros=s.substr(1);

    subseq(ros,ans);

    subseq(ros,ans+ch);

}

int main(){

    subseq("ABC","");

}

## 15. genrate substring with ascii code

/\*

"" 3 call at a time one to incluce one to exclude and one to add ascii value

"B"

"66"

"A"

"BA"

"66A"

"B65"

"6665"\*/

## 16.generate all substring of a string

void subseq(string s,string ans){

    if(s.length()==0){

        cout<<ans<<endl;

        return;

    }

    char ch=s[0];

    string ros=s.substr(1);

    subseq(ros,ans);

    subseq(ros,ans+ch);

    subseq(ros,ans+to\_string(int(ch)));

}

int main(){

    subseq("AB","");

}

## 17.permutation all the string

QUES-ABC

SOLUTION-

ABC

ACB

BAC

BCA

CAB

CBA

void permute(string s,int l,int r){

    if(l==r){

        cout<<s<<endl;

        return;

    }

    for(int i=l;i<=r;i++){

        swap(s[l],s[i]);

        permute(s,l+1,r);

        //backtracking

        swap(s[l],s[i]);

    }

}

void permute2(string s,string ans){

    if(s.length()==0){

        cout<<ans<<endl;

        return;

    }

    for(int i=0;i<s.length();i++){

        string ros=s.substr(0,i)+s.substr(i+1);

        permute2(ros,ans+s[i]);

    }

}

int main(){

    string str="ABC";

    int n=str.size();

    permute2(str,"");

    permute(str,0,n-1);

}

## 18.write a prpgram to check num is palindrome

mehtod 1 number to string

method 2

#include <math.h>

int ReverseNumber(int num){

    if(num<10) return num;

    int digit = (int)log10(num); // to calculate number of digit

    return (((num % 10)\*pow(10,digit))+ReverseNumber(num/10)); // pow func cause depending upon place value no of 10s is multiplied

}

bool PalindromeNumber(int num){

    if(num<10) return true;

    else if (ReverseNumber(num)==num) return true;

    else return false;

}

// method 3

bool f(int num,int \*temp){ //temp if refrence any change in temp is stored

    if(num<10){

        bool ans = (num == ((\*temp)%10));

        (\*temp)/=10;

        return ans ;

    }

    bool result = (f(num/10, temp) and (num%10)==((\*temp)%10));

    (\*temp)/=10;

    return result;

}

## 19.given a number n.find the inc sequence from 1 to n without using loop

void seq(int n){

    if (n==0) return ;

    seq(n-1);

    cout<<n<<" ";

}

## 20.given a number n and a value k, print k multiples of n

void multiple(int n, int k){

    if(k==0) return;

    multiple(n,k-1);

    cout<<n\*k<<endl;

}

## 21.given a number n.find the sum of natural number till n but with alternative size

int sum1(int n){

    if(n==1) return 1;

    int sum;

    if ((n%2)==0){

        sum=sum1(n-1) - n;

        return sum;

    }

    if((n%2)!=0){

        sum=sum1(n-1) + n;

        return sum;

    }

}

## 22.given two numbers x and y. find the greatest common divisor of x and y using recursion

### using iretation

lowest of x and y suppose x

for(i=x;i<0,i++){

    if(x%i==0 and y%i==0)

    cout<<i;

    break;

}

### using recursion

int hcf(int n1, int n2) {

    if(n2<n1){

        if (n2 != 0) return hcf(n2, n1 % n2);

        else return n1;

    }

    if(n1<n2){

        if (n1 != 0) return hcf(n1, n2 % n1);

        else return n2;

    }

}

### method 2 -Euclid’s algorithm

if we subtract a smaller number from a larger number,(we can reduce the larger number) byt the gcd/hcf remains same

eg 54,78 = 18

(54,78-54) = (54,18) = (18,54-18) = (18,36) = (18,36-18) = (18,18)

int hcf2(int x, int y){

    if(x==y) return x;

    if(x>y) return hcf2(x-y,y);

    if(y>x) return hcf2(x,y-x);

}

## 23.calc Armstrong

int S\_cubeOfEle(int n){

    if(n<10) return n;

    return (pow((n%10),3)+S\_cubeOfEle(n/10));

}

bool armstrong(int num){

    if(S\_cubeOfEle(num)==num) return 1;

    else return 0;

}

## 24.given an array n and value x check wheather x exist or not

bool check\_ele(int n[],int l, int x, int i){

    if(i==l) return false;

    return (n[i]==x) || check\_ele(n,l,x,i+1);

}

## 25.Frog problem

QUES

There are N stones, numbered 1,2,...,N. For each i (1<=i<=N), the height of Stone i is hi. There is a frog who is initially on Stone 1. He will repeat the following action some number of times to reach Stone N:

If the frog is currently on Stone i, jump to Stone i+1 or Stone i+2. Here, a cost of hi - hj is incurred, where j is the stone to land on.

Find the minimum possible total cost incurred before the frog reaches Stone N.

SOLUTION

we dont know the the complete cost of 1 to n

we know only 1 se 2 ki yah 1 se 3 ki therefore issi se recursive sol banega

1 se 2 yah 1 se 3 me joh chota hai choose take after chosing suppose 2 chota hai 2 se 3 chota hai yah 2 se 4

aur yeahi mai ulta chalo 4 se 2 and then 2 se 1

f(stone ki array height,length,index) = return mini cost to reach fromith to nth

f()-> 1.f(h,n,i+1) + hi-h(i+1) #note= f(h,n,i+1) will given min cost from n to i+1 then from i+1 tto i i will add the diffrence if there hight

2.f(h,n,i+2) + hi-h(i+2)

inn dono path me se joh min hoga voh ans hoga

int frog(int h[],int n,int i){

    if(i==n-1) return 0;

    if(i==n-2) return frog(h,n,i+1)+ abs(h[i]-h[i+1]);

    return min(frog(h,n,i+1)+abs(h[i]-h[i+1]),frog(h,n,i+2)+abs(h[i]-h[i+2]));

}

## 26.Tower of hanoi

we need to shift the tower from a to c such that we can shift only one block at the time and in c only small block should be at the big block

A is the source B is the helper rod and C is the destination

void towerOfHanoi(int n,char src,char helper,char dest){

    //base case

    if(n==0) return;

    //source to helper

    towerOfHanoi(n-1,src,dest,helper);

    cout<<"Move disk " << n << " from rod " << src << " to rod " << dest << endl;

    //helper to destination

    towerOfHanoi(n-1,helper,src,dest);

}

int main(){

    int n=3;

    towerOfHanoi(n,'A','B','C');

}