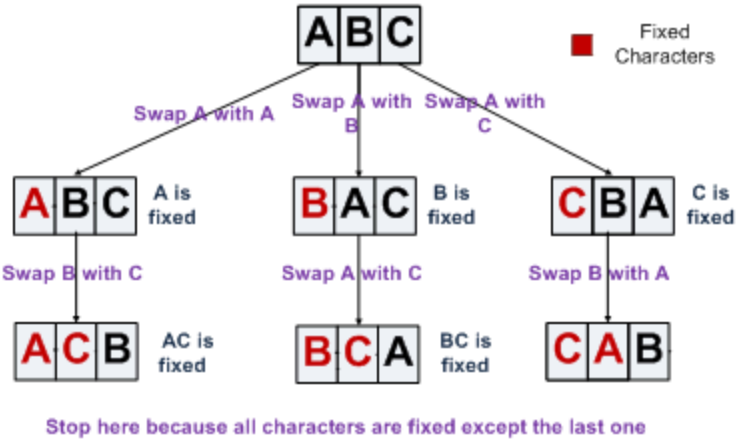
Write a C program to print all permutations of a given string

August 2, 2009

A permutation, also called an “arrangement number” or “order,” is a rearrangement of the elements of an ordered list S into a one-to-one correspondence with S itself. A string of length n has n! permutation.
Source: Mathword(http://mathworld.wolfram.com/Permutation.html)

Below are the permutations of string ABC.
ABC, ACB, BAC, BCA, CAB, CBA

Here is a solution using backtracking.



Recursion Tree for Permutations of String "ABC"

```
# include <stdio.h>

/* Function to swap values at two pointers */
void swap (char *x, char *y)
{
    char temp;
    temp = *x;
    *x = *y;
    *y = temp;
}

/* Function to print permutations of string
   This function takes three parameters:
   1. String
   2. Starting index of the string
   3. Ending index of the string. */
void permute(char *a, int i, int n)
{
   int j;
   if (i == n)
     printf("%s\n", a);
   else
   {
        for (j = i; j <= n; j++)
       {
          swap((a+i), (a+j));
          permute(a, i+1, n);
          swap((a+i), (a+j)); //backtrack
       }
   }
}

/* Driver program to test above functions */
int main()
{
```

```
    char a[] = "ABC";
    permute(a, 0, 2);
    getchar();
    return 0;
}
```

Algorithm Paradigm: Backtracking
Time Complexity: O(n*n!)

80 comments so far

1. *Avinash* says:
   February 13, 2012 at 2:38 AM

```
permute(int input[])
{
    int n=size(input[]);
    int out[]=malloc(sizeof(input[]));
    int used[]=malloc(sizeof(input[]));
    for(int i=0;i<size(used[]);i++)
    {
        used[i]=0;
    }
    dopermute(input,0,n,out,used);
}

void dopermute(int input[],int level, int size, int out[],int used[])
{
    If(level>=size)
    {
        for(int i=0;i<size(out);i++)
        {
            printf("%c",out[i]);
        }
        return;
    }

    for(j=0;j<size;j++)
    {
        If used[i] continue;
        used[j]=1
```

```
            out[level]=arr[j];
            dopermute(input,level+1,size,out,used);
            used[j]=0;
        }
    }
```

Reply

2. *niak* says:
February 11, 2012 at 11:31 AM

number is {1,2,3,4,5,6}it will print as by rotating 1 place it will look likjs 6,1,2,3,4,5

Reply

3. *guru* says:
February 7, 2012 at 8:59 PM

```
ArrayList<String> permutationsOf(String s) {
    ArrayList<String> result = new ArrayList<String>();

    if (s.length() == 1) {

        result.add(s);
        return result;
     }
else {

        char first = s.charAt(0);
        String rest = s.substring(1);

              ArrayList<String> simpler = permutationsOf(rest);

        for (String permutation : simpler) {

            ArrayList additions = insertAtAllPositions(first, permutation);

            result.addAll(additions);
        }
        return result;
    }
}
```

Reply

4. *ckernel* says:
February 5, 2012 at 9:19 AM

Hi Geeksforgeeks,

Neat Code. I just want to learn how you devised the backtracking algorithm for this problem?

//I mean I wanted to know the explanation of the tree given above.

-ckernel

Reply

5. *omair hassan* says:
   January 31, 2012 at 3:23 PM

   hi

   can any one suggest me good tutorial on c programming, i m new in programming , these days i m reading oop in c++ by lafore , please suggest me something that really helps me .. from the skretch ....

   Reply
   - *addy* says:
     February 7, 2012 at 3:01 PM

     You can use use thinking in c++ Vol 1 By Bruce Eckel

     http://www.digilife.be/quickreferences/books
     /thinking%20in%20c++,%20volume%201,%202nd%20edition.pdf

     Reply
6. *omair hassan* says:
   January 30, 2012 at 1:41 AM

   hi\
   i copy the code in my complier but its not working
   its prompt the following errors
   1. 29 line , canno convert 'char * ', to 'char'
   2. 29 line. type mismatch in parameter 'a' in call to ' permute(char, int
   3, 32 line , 'a' is assigned a value that is never used

   please help me i m new in programming

   Reply
   - *GeeksforGeeks* says:
     January 30, 2012 at 11:27 AM

     @omair hassan: The program has been tested with more than compilers and it works fine. See this for a sample run that works. Could you let us know the compiler you used and the exact code that you tried?

     Reply
     - *omair hassan* says:
       January 30, 2012 at 5:42 PM

       i am using turbo c ++

       Reply
       - *omair hassan* says:
         January 30, 2012 at 6:50 PM

         thank you so much for the reply . the codes runs finally at char swap (*x, *y) i was using char swap (a,b)

         can u tell me what was the mistake , although i didn't understand the code
         please help me ....

Reply

- *geek4u* says:
  January 30, 2012 at 8:52 PM

  Following tutorial on pointers and function may help.

  http://www.cs.cf.ac.uk/Dave/C/node10.html

- *omair hassan* says:
  January 30, 2012 at 8:55 PM

  that's so helpful thanks alot dude

- *Sandeep* says:
  January 31, 2012 at 6:57 PM

  The C book (http://publications.gbdirect.co.uk/c_book/) seems to be a good resource for C.

- *omair hassan* says:
  February 1, 2012 at 9:11 PM

  @sandeep

  thank u so much dude

7. *Jailbreak and unlock your iPhone* says:
   January 15, 2012 at 1:40 PM

```
<?php

function sort_string($str){
    $arr = NULL;
    $len = strlen($str);
    for($i=0; $i<$len; $i++){
        $arr[] = $str[$i];
    }
    sort($arr);
    $str = "";
    for($i=0; $i<$len; $i++){
        $str = $str . $arr[$i];
    }
    return $str;
}

function permute($str){
    //base cases
    if ($str == NULL){
        return "";
    }
    $len = strlen($str);
    if ($len == 1){
        return $str;
    }
```

```
    else if ($len == 2){
        return $str ."," . $str[1] . $str[0];
    }
    // the recursion
    else {
        $ans = "";
        for($i=0; $i<$len; $i++){
            // currenct char
            $char = $str[$i];
            // the string without the selected char
            $str_b = substr($str,0,($i-1 < 0 ? 0 : $i)) . substr($str,$i+1, $
            //build all the sub-permutations without char
            $sub_perm = permute($str_b);
            //now add char
            $ans = $ans . add_first_char($sub_perm, $char);
        }
        return $ans;
    }
}

function add_first_char($str, $char){
    $ans = "";
    $items = explode (',',$str);
    $len = sizeof($items);
    for($i=0; $i<$len; $i++){
        $ans = $ans . "," .  $char . $items[$i];
    }
    // remove the first ","
    return $ans;//
}

$filename = $argv[1];
$handle = fopen($filename, "rb");
$line = "";
while (!feof($handle)) {
  $line = fgets($handle, 8192);
  $line = trim($line);
  if ($line != NULL) {
      $line = sort_string($line);
      $ans = permute($line);
      //remove first "," from the final response
      $ans = substr($ans,1,strlen($ans)-1);
      echo trim($ans) . "\n";
  }
}
fclose($handle);
return 0;

?>
```

Reply

8. *Venkatesh* says:
   January 12, 2012 at 1:43 PM

http://www.programmerinterview.com/index.php/recursion/permutations-of-a-string/

Reply

9. *amit* says:
December 12, 2011 at 5:39 AM

http://n1b-algo.blogspot.com/2009/01/string-permutations.html

Reply

10. *pratik* says:
December 7, 2011 at 11:46 AM

Please explain how the time complexity is O(n!)??

```
/* Paste your code here (You may delete these lines if not writing code) */
```

Reply

- *PRATEEK* says:
December 29, 2011 at 1:27 AM

see the for loop inside the func..permute where permute is called again and again 1 less than current n times
ie, n*n-1*n-2*.....*1=O(n!)
or basically we could understand like this
eg ankit, we have find all possible permutations of the name ankit
as length of ankit is 5 thereforeforloop is called 5 times and then4 3 2 1 and so on as maximum permutations of agiven word of length n is n! therefore we can simply say the complexity is O(n!).....

Reply

11. *navyasree* says:
December 6, 2011 at 10:16 PM

hii..
i want a program to print the first letter of given sentence of each word and at the end the last word of given sentence also printed..
for ex:
the given string is: hii hello how are you
output:hhhayyou
like this... pls anybody tel me the code...

Reply

- *coder* says:
December 7, 2011 at 1:51 PM

1.Read the line.
2.Tokenize the string.
3.For every token take the char at 0th position and store it in a another string or concatenate to your result string.
4.Repeat step 3 till tokens are there.

Reply

- *Arpit Gupta* says:

December 25, 2011 at 3:41 PM

```
#include<stdio.h>
#include<string.h>
void print(char *s)
{   int i,word_start_index=0;
    printf("%c",s[0]);
    for(i=0;s[i]!='\0';i++)
    {   if(s[i]==32)
        {   printf("%c",s[i+1]);
            word_start_index=i+1;
        }
    }
    printf("%s\n",s+word_start_index);
}
int main()
{   char *s="hii hello how are you";
    print(s);
    return 0;
}
```

Reply

12. *pinku* says:
November 24, 2011 at 5:41 PM

& plzz give a flow chart also

Reply

13. *pinku* says:
November 24, 2011 at 5:38 PM

plzz reply sharp.....write a program in c to print all posible permutations of abcde

```
/* Paste your code here (You may delete these lines if not writing code) */
```

Reply

14. *sandy880* says:
November 24, 2011 at 2:09 PM

```
#include <stdio.h>
#define SIZE 3
int main(char *argv[],int argc)
{
char list[3]={'a','b','c'};
int i,j,k;
for(i=0;i<SIZE;i++)
for(j=0;j<SIZE;j++)
for(k=0;k<SIZE;k++)
if(i!=j && j!=k && i!=k)
printf("%c%c%c\n",list[i],list[j],list[k]);
return(0);
}
```

Reply

15. *mahendhar rao* says:
October 19, 2011 at 9:07 AM

Awesome website!!!

Reply

16. *shantanu bhawre* says:
October 15, 2011 at 11:31 PM

All are correct but i want the otput like
if the string is "abcd"
then the combinatio will
bcda
cdab
dabc
abcd

like that can u tell me

Reply

- *sura* says:
December 20, 2011 at 1:35 PM

```
public static void main(String[] args)
    {
        char[] input = "abcd".toCharArray();
        int j=0;

        for(int i=0;i<input.length;i++)
        {
            int count=0;
            j=i;
            while(count<input.length)
            {
                count++;
                System.out.print(input[j]);
                j=(j+1)%input.length;
            }
            System.out.println();
        }
    }
```

Reply

17. *pps* says:
October 15, 2011 at 12:16 PM

nice

Reply

18. *ravi* says:
September 17, 2011 at 8:22 AM

really good site...for every question there is explanation

Reply

- *ror* says:
  September 28, 2011 at 3:59 AM

  but how about if a want a permutation list of something like 1238596?

  Reply

19. *Amitabh* says:
    September 12, 2011 at 12:14 AM

    plz get me a program for this output

    1
    121
    12321
    121
    12121

    Reply

20. *Ravinder Bishnoi & Yugarsi Ghosh* says:
    August 14, 2011 at 11:42 PM

    Guys,here is a very simple solution....

```
#include
#include
#include
using namespace std;
char a[10];
void func(char *a,int j=0)
{
if(j==strlen(a))
{
printf("%s\n",a);
}
char ch='';
for(int i=j;i<strlen(a);i++)
{
char t;
if(ch==a[i])
continue;
else
{
ch=a[j];
t=a[i];
a[i]=a[j];
a[j]=t;
func(a,j+1);
}
}
}
int main()
```

```
{
char a[10];
printf("Enter a String\n");
scanf("%s",&a);
printf("Permutations of string\n");
func(a);
getch();
return 0;
}
```

Reply

- *Ravinder Bishnoi & Yugarsi Ghosh* says:
  August 15, 2011 at 12:25 AM

  above program was not posted correctly...
  use the new one...
  it works when all elements of string are distinct otherwise some permutations will be repeated.
  headers are iostream,string and conio.h.

```
#include
#include
#include
using namespace std;
char a[10];
void func(char *a,int j=0)
{
if(j==strlen(a))
{
printf("%sn",a);
}
for(int i=j;i<strlen(a);i++)
{
char t;
t=a[i];
a[i]=a[j];
a[j]=t;
func(a,j+1);
}
}
int main()
{
char a[10];
printf("Enter a Stringn");
scanf("%s",&a);
printf("Permutations of stringn");
func(a);
getch();
return 0;
}
```

  Reply

- *Venki* says:

September 12, 2011 at 10:32 PM

@Ravinder & @Yugarsi, Here is your logic. How is it better, could you explain your view?

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

void permute(char *a, int j = 0)
{
    int len = strlen(a);

    if(j == len)
    {
        cout << a << endl;
    }

    for(int i = j; i < len; i++)
    {
        char t = a[i];
        a[i] = a[j];
        a[j] = t;

        permute(a, j+1);
    }
}

int main()
{
    char a[] = "ABCD";
    permute(a);
    return 0;
}
```

This kind of programs runs like O(f(n) * n!) or O(f() * 2^n) where is f(n) is polynomial in n. Any algorithm must generate all possible permutations.

Reply

21. *amit khoth* says:
August 12, 2011 at 11:53 PM

@geeksforgeeks correct this code.for same type of word it is not working.

Reply

- *GeeksforGeeks* says:
August 13, 2011 at 2:20 AM

@amit khoth: Could you please provide a sample string for which it didn't work?

Reply

  - *amit khoth* says:
August 13, 2011 at 10:59 AM

when two character are similar in a word.like aatif,aaaj etc,

Reply

- *Christian* says:
  February 6, 2012 at 1:52 PM

  Well, what do you think it should do? It's probably undefined for input like that.

  Reply

22. *Anand* says:
    June 7, 2011 at 9:23 AM

    http://anandtechblog.blogspot.com/2011/05/arra-y-problem.html

    Reply
23. *Sam* says:
    June 5, 2011 at 7:12 PM

    Great post!

    Nice solution here as well:

    http://www.programmerinterview.com/index.php/recursion/permutations-of-a-string/

    Reply
24. *rimu.nitrkl* says:
    March 19, 2011 at 1:23 AM

    The code mentioned above doesnt work correctly, if there are any repeated characters in the string, it would unncessarily print some strings multiple times ...

    Working code for the same, which takes care of repeated chars as well:

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

int printPermutations(char *str,int size, int pos)
{
     int i;
     int total=0;

     if(pos==(size-1))
     {
                    puts(str);
                    return 1;
     }

     total+=printPermutations(str,size,pos+1);
     for(i=pos+1;i<size;i++)
     {
                         int j;
                         for(j=pos;j<i;j++)
                                          if(*(str+j)==*(str+i))
                                                           break;

                         if(j==i)
```

```
                {
                        char tmp=*(str+pos);
                        *(str+pos)=*(str+i);
                        *(str+i)=tmp;

                        total+=printPermutations(str,size,pos

                        tmp=*(str+pos);
                        *(str+pos)=*(str+i);
                        *(str+i)=tmp;
                }
        }

        return total;
}

int main()
{
    char str[100];
    int size,total;

    printf("Enter the string: ");
    gets(str);

    size=strlen(str);

    printf("\n\nAll permutations of the input string are:\n");
    total=printPermutations(str,size,0);

    printf("\n\nThe total number of permutations of the given string is %d",t

    getch();
    return 0;
}
```

Reply

- *ricky* says:
  April 19, 2011 at 7:34 PM

  @rimu.nitrkl...Could You Please Through Some More Light ..I mean Can you write down the algorithm..step by step..its really nice way to programming that you have done..also..i think you TC is O(n^2)..isn't it..??

  Waiting for your Explanation..??? Reply ASAP

  Reply

25. *crash* says:
    March 3, 2011 at 9:45 PM

    i want a all combos of 7 letters
    should i hav to use char

    Reply

26. *Ahul* says:

February 18, 2011 at 9:34 PM

How the control again reaches to permute fn evenafter the i becomes 2

Reply

27. *Matrix* says:
February 6, 2011 at 10:03 PM

the first program gives repeated permutation in case of 122
how to correct it
plz help ??
in case of 122,it gives
122
212
221
212 //repeated
221 //repeated

Reply

28. *world is round* says:
February 2, 2011 at 7:33 PM

Another way to do this : http://layerinside.blogspot.com/2011/02/permutation-of-string.html

Reply

29. *Shashank Mani Narayan* says:
January 30, 2011 at 7:50 PM

Hey guys..I know Most of You Stuck with this Question..The 1st Backtracking approach..seems to typical..for understanding [points of view.....here i am posting a excellent so0lution fro problem hope this will help you ..lot..

Explanation

Let's assume a given string S represented by the letters A1, A2, A3, ..., An
To permute set S, we can select the first character, A1, permute the remainder of the string to get a new list. Then, with that new list, we can "push" A1 into each possible position.
For example, if our string is "abc", we would do the following:
1. Let first = "a" and let remainder = "bc"
2. Let list = permute(bc) = {"bc", "cd"}
3. Push "a" into each location of "bc" (--> "abc", "bac", "bca") and "cb" (--> "acb", "cab", "cba")
4. Return our new list

```
import java.util.*;
class Permutation
{
public static ArrayList getPerms(String s)
{

 ArrayList permutations = new ArrayList();

 if (s == null)
 { // error case
```

```
    return null;
  }
  else if (s.length() == 0)
  { // base case
  permutations.add("");
  return permutations;
  }

  char first = s.charAt(0); // get the first character
  String remainder = s.substring(1); // remove the first character
  ArrayList words = getPerms(remainder);

  for (String word : words)
  {

     for (int j = 0; j <= word.length(); j++)
     {
       permutations.add(insertCharAt(word, first, j));
     }

   }
        return permutations;

   }

   public static String insertCharAt(String word, char c, int i)
   {
     String start = word.substring(0, i);
     String end = word.substring(i);

      System.out.println("start=" + start  + "\t c=" + c + "\t end=" + end );
     return start + c + end;
   }

   public static void main(String a[])

   {
      ArrayList perm = new ArrayList();

      perm=getPerms("abc");

     //for(String ele:perm)
           //System.out.println(ele);
   }
}
```

Compile: javac Permutation.java
Run: java Permutation

/*output analysis
start= c=c end=
start= c=b end=c
start=c c=b end=
start= c=a end=bc

start=b c=a end=c
start=bc c=a end=
start= c=a end=cb
start=c c=a end=b
start=cb c=a end=
*/

Reply

30. *Sai Ganesh* says:
December 18, 2010 at 12:31 PM

My solution:

```
#include<stdio.h>
void printer(int *a,int n){
    int i;
    for(i=0;i<n;i++)printf("%d",a[i]);printf("\n");
}
/*recursive permute function
Semantics:
    a - array which holds the present permutation of size n

    k - present slot in the array a. we have a permutation when k==n
    if k < n , we are still building our permutation.

    We can think of the permutation result as
    a row of n slots, holding a number between 1 and n

    (if we want permutations from another array b, we can think of
    it as holding b[1] to b[n] ..the logic is the same)

    if we don't want them to repeat, we need to maintain
    state of our computations in the array called 'selected'.

 */
void permute(int *a,int n,int k,int *selected){
    if(k==n){
        printer(a,n);
        return;
    }
    int i;
    for(i=0;i<n;i++){
        if(!selected[i]){
            a[k]=i+1; //alternately, a[k]=f(i) , here f(i)=i+1.
            //essentially, the i'th value from a list.
            selected[i]=1;
            permute(a,n,k+1,selected); //fill up from the next index
            selected[i]=0; //backtrack
        }
    }
}
int main(void){
    int a[10]={0};
    int i;
```

```
    int selected[10]={0};
    permute(a,5,0,selected);
    return 0;
}
```

Advantages:
1. code is clear and the printer() function can be replaced by
a process() function which is 'streamed' one permutation at a time

2. It can also be used to permute values from a function.
we are just storing the indices.

3. Can be extended to print permutations with repetition:
make the if condition always evaluate to true
like (if(1 || !selected[i])) ...

Reply

- *Emran* says:
  April 8, 2011 at 3:28 PM

  this is a really clean implementation, and i like that it lists in lexicographic order.

  question: what if i had an application where i could reduce run time by avoiding some permutations...
  like if i wanted to enumerate every other permutation, or only the first (n!)/2 permutations?

  in particular, ive reached a case in a problem im solving where 1,2,3,4 is the effective equivalent of 4,3,2,1...
  so i want to remove those from being evaluated; but im not sure how to group out such subsets...

  Reply
- *seeker7* says:
  April 16, 2011 at 10:32 AM

  This code is plagiarised from Programming interview exposed
  I do not intend to Offend you ,but you should have mentioned the source of your code .

  Reply

31. *Phani Deepak* says:
    October 25, 2010 at 8:13 PM

    Just another improvement.
    The function prototype can be just permute(char *a, int size);

    Modification would be:
    instead of swapping every element in loop with first element,
    we can swap it with last element and call permute(a, size-1);

    what do you say?

    Reply
32. *cod3r* says:
    September 22, 2010 at 6:06 AM

Hey,
how would do it for strings with repeated string

Reply

- *cod3r* says:
  September 22, 2010 at 6:06 AM

  Hey,
  how would do it for strings with repeated **char**

  Reply

  - *Divye* says:
    July 27, 2011 at 7:29 PM

    See this post to understand the logic behind printing all permutations without duplicates (even when duplicate characters are present in the input).

    http://www.divye.in/2011/06/printing-all-permutations-of-string.html

    Reply

33. *Anshul* says:
    August 23, 2010 at 5:59 PM

    I think this will work. I ran this and its working:

```
private void swap(ref char a, ref char b)
        {
            if (a == b) return;
            a ^= b;
            b ^= a;
            a ^= b;
        }

        private void go(char[] list, int k, int m)
        {
            int i;
            if (k == m)
            {
                Console.Write(list);
                Console.WriteLine(" ");
            }
            else
                for (i = k; i <= m; i++)
                {
                    if (list[k] == list[i] && k != i) continue;
                    swap(ref list[k], ref list[i]);
                    go(list, k + 1, m);
                    swap(ref list[k], ref list[i]);
                }
        }
```

    Only difference is in Swap function, So swap is also checking if swapping characters are equivalent.

    Reply

- *stackoverflow* says:
  January 27, 2011 at 12:36 PM

  copied well from stack overflow ..... nice!!

  Reply

34. *Anshul* says:
    August 23, 2010 at 5:59 PM

    For the main code given on this page, I think for string = “aaa” it will print “aaa” several times instead of just once. To remove this error, do the following modification:

```
for (j = i; j <= n; j++)
{
    if (A[i] == A[j] && i != j)
        continue;
    swap((a+i), (a+j));
    permute(a, i+1, n);
    swap((a+i), (a+j)); //backtrack
}
```

    The highlighted line is not preventing print of same "AAA". I think we need to check for same string and should not execute in case of same string.

    Reply

35. *Ankul* says:
    August 13, 2010 at 1:58 AM

    For the main code given on this page, I think for string = "aaa" it will print "aaa" several times instead of just once. To remove this error, do the following modification:

```
for (j = i; j <= n; j++)
       {
         if (A[i] == A[j] && i != j) continue;
          swap((a+i), (a+j));
          permute(a, i+1, n);
          swap((a+i), (a+j)); //backtrack
       }
```

    Reply

    - *Cracker* says:
      November 17, 2011 at 1:40 AM

      not working for string "agra"

      Reply

      - *Cracker* says:
        November 17, 2011 at 1:55 AM

        output is --

        agra
        agar
        arga

arag
aarg
aagr
gara
gaar
graa
rgaa
raga
raag
raag
raga

Reply

36. *seeker7* says:
July 31, 2010 at 9:00 AM

hw abt this:
rotate the given array n-1 times
then reverse the intial array and again rotate it n-1 times!

woud'nt it be better than using recursion?

Reply

- *Ravinder* says:
November 24, 2010 at 1:03 PM

It doesn't provide all solutions!!?
instead of n! permutations, it just gives 2n

correct me in case of misinterpretation

Reply

37. *Anshu Bansal* says:
July 12, 2010 at 10:23 PM

Would this work for 'N' number of characters in String?

Reply

38. *rscrbv* says:
July 4, 2010 at 2:10 AM

here is my code, if anybody likes

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

void ror(char str1[],int a,int b)
{
    char temp=str1[a];
    int j;
    for(j=a;j<b;j++)
    {
        str1[j]=str1[j+1];
```

```
    }
    str1[b]=temp;
}

void perm(char str[],int f,int l)
{
     int i;
     char str1[5];
     strcpy(str1,str);
     if(f==l)
     {
        printf("%s\t",str1);
        return;
     }
     for(i=f;i<=l;i++)
     {
        perm(str1,f+1,l);
        ror(str1,f,l);
     }
}

int main()
{
      perm("vikas",0,4);
      getchar();
}
```

Reply

- *shashank* says:
  January 3, 2011 at 11:47 AM

  @rscrbv
  @geeksforgeek
  ..could plz tell me how ur code is working..plz help le out that how the control is tranferring..take string abc & plz show ur flow control at for every combination...i really need help in dis..program...reply asap.

  Reply

  - *wgpshashank* says:
    January 6, 2011 at 8:31 PM

    @rscrbv
    @geeksforgeek
    ..could plz tell me how ur code is working..take string abc & plz show ur flow control at for every combination…i really need help in dis..program…reply asap.

    Reply

- *seeker7* says:
  April 16, 2011 at 10:27 AM

  why does not any one explain this well-loved code
  thanks!

Reply

- *vivek singh* says:
  May 13, 2011 at 6:40 PM

  well done

  Reply

39. *remo* says:
    May 22, 2010 at 4:22 PM

    what will done when call Function permute(a, i+1, n);

```
for (j = i; j &lt;= n; j++)

      {

         swap((a+i), (a+j));

         permute(a, i+1, n);

         swap((a+i), (a+j)); //backtrack

      }
```

    Reply

40. *Sven Forstmann* says:
    April 8, 2010 at 2:40 PM

    Simple and non-recursive, lexically correct output.

```
std::string default = "Hallo";

int perm=1, digits=default.size();
for (int i=1;i<=digits;perm*=i++);
for (int a=0;a<perm;a++)
{
    std::string avail=default;

    for (int b=digits,div=perm;b>0; b--)
    {
        div/=b;
        int index = (a/div)%b;
        printf("%c", avail[index] );
        avail.erase(index,1) ;
    }
    printf("\n");
}
printf("permutations:%d\n",perm);
```

    (c) Sven Forstmann

    Reply

    - *kartik* says:
      April 10, 2010 at 8:27 PM

@Sven Forstmann: Can you please write algorithm for the above code?

Reply

- *bala* says:
  January 5, 2011 at 8:46 AM

  This code does not work for non-unique set of characters. But works for unique set of characters.

  Reply

- *anom* says:
  January 22, 2011 at 11:18 AM

  You are assuming the string contains no dup chars. However, your sample string has two 'l's. Your own sample doesn't work.

  Reply

41. *Sandeep* says:
    March 21, 2010 at 10:34 PM

    @amit: The diagram shows recursive execution of permute().

    For i = 0 and j = 0
    A is fixed at first place using below line

    ```
    swap((a+i), (a+j)) /*A is swapped with A*/
    ```

    Then all the permutations of BC (sub-string after A) are printed

    ```
    permute(a, i+1, n);  /*Call permute for BC  with i = 1  */
    ```

    Finally swap the characters back

    ```
    swap((a+i), (a+j)) /*A is swapped with A*/
    ```

    For i = 0 and j = 1
    B is fixed at first place using below line

    ```
    swap((a+i), (a+j)) /*A is swapped with B*/
    ```

    Then all the permutations of BC (sub-string after A) are printed

    ```
    permute(a, i+1, n);  /*Call permute for AC  with i = 1 */
    ```

    Finally, swap the characters back

    ```
    swap((a+i), (a+j)) /*B is swapped with A*/
    ```

    For i = 0 and j = 2
    C is fixed at first place using below line

    ```
    swap((a+i), (a+j)) /*A is swapped with C*/
    ```

Then all the permutations of BC (sub-string after A) are printed

```
permute(a, i+1, n);  /*Call permute for BA with i = 1 */
```

Finally, swap the characters back

```
swap((a+i), (a+j)) /*C is swapped with A*/
```

For i = 1, second character is swapped one by one with the other characters (after second character). Same way is continued for i = 2, 3..

Reply

42. *amit* says:
March 20, 2010 at 7:17 PM

@geeksforgeeks
can u explan how this code works.

Reply

43. *Ved* says:
January 26, 2010 at 7:15 PM

The complete code is at :
http://techpuzzl.wordpress.com/2010/01/11/string-permutation/

Reply

44. *Srinivas Iyengar* says:
November 15, 2009 at 11:17 PM

Ok.
We can call this function like perm("",s) where s is the string which has to be permuted.
I am not sure how to highlight the code in the Comment Section. I hope the code is readable enough.

Reply

45. *geeksforgeeks* says:
November 15, 2009 at 11:11 PM

@Srinivas Iyengar: Could you add more colors to the code given below, please? What should be prefix and suffix for the first call of permute() in main()?

Reply

46. *Srinivas Iyengar* says:
September 8, 2009 at 11:07 AM

```
/*Similar Recursive Code for Permutation of a String keeping track of
   prefix and suffix*/
void permute(string prefix,string suffix)
{
   /* We can print prefix here for generating power set and simply
       write return in the following if case.*/
  if(suffix.size()==0)
    printf("%s \n", prefix);
  else
   for(int i=0;i < suffix.size();i++)
```

```
        permute(
                 prefix+suffix[i],
                 suffix.substr(0,i)+suffix.
                 substr(i+1, suffix.size())
                );
    }
```

Reply

Comment

Name (*Required*) Email (*Required*) Website URI

Your Comment (Writing code? please paste your code between sourcecode tags)

```
[sourcecode language="C"]
/* Paste your code here (You may delete these
lines if not writing code) */
[/sourcecode]
```

Have Your Say

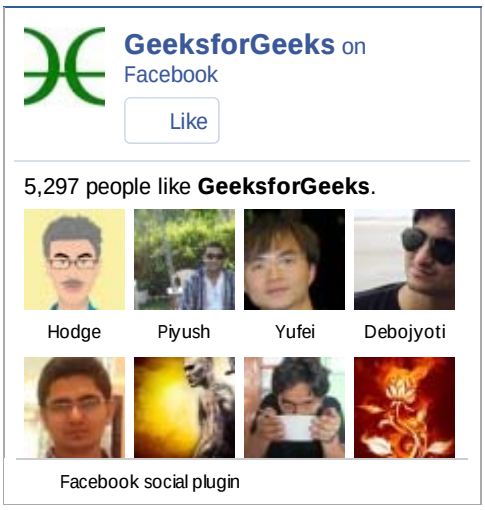
Notify me of followup comments via e-mail

Subscribe without commenting
E-Mail: Subscribe

- Search

- # Popular Tags

  - GATE
  - Java
  - Dynamic Programming
  - Divide & Conquer
  - Backtracking
  - Pattern Searching
  - Operating Systems
  - Recursion



- # Forum Latest Discussion

  - Google interview question
    Last Post By: Gautam
    Inside: Interview Questions
  - Recursion concept
    Last Post By: kartik
    Inside: Miscellaneous
  - knights tour
    Last Post By: kartik
    Inside: Interview Questions
  - Microsoft Interview Question for Software Testing (2-5 Years) about Data Structu [...]
    Last Post By: Aashish Barnwal
    Inside: Interview Questions
  - Amazon Interview Question for Software Engineer/Developer about Trees
    Last Post By: camster
    Inside: Interview Questions
  - how to debug the c code.....?
    Last Post By: DJ
    Inside: Interview Questions
  - Count of paths
    Last Post By: codinglearner
    Inside: Interview Questions
  - what is the output ?? and why
    Last Post By: kartik
    Inside: C/C++ Programming Questions

- # Popular Posts

  - Sorted Linked List to Balanced BST
  - Tree traversal without recursion and without stack!
  - All permutations of a given string
  - Memory Layout of C Programs
  - Longest substring without repeating characters
  - Structure Member Alignment, Padding and Data Packing

- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Understanding “extern” keyword in C
- Median of two sorted arrays

Forum Categories

- Interview Questions
- C/C++ Programming Questions
- Algorithms
- Trees specific questions
- Linked List specific questions
- Multiple Choice Questions
- Object oriented queries
- GPuzzles
- Operating Systems
- Miscellaneous
- Java specific Questions
- Perl specific Questions

Add Interview Questions

If you have attended an interview recently or have good interview questions to share with the fellow geeks and get the best solution, please add your questions to Add Interview Questions page

Recent Comments

- kartikaditya on Length of the longest substring without repeating characters
- kartikaditya on Find the maximum element in an array which is first increasing and then decreasing
- Avinash Kumar on Write a function to reverse a linked list
- Avinash Kumar on Write a function to reverse a linked list
- kartikaditya on Minimum number of jumps to reach end
- kartikaditya on Find the largest BST subtree in a given Binary Tree
- kartikaditya on Find the largest BST subtree in a given Binary Tree
- Frederic on Print all interleavings of given two strings