ជំពូកទី ៨

អំពី String

ទោះបីយើងបានប្រើ អញ្ញាតប្រភេទ char និង arrays ដែលមានតម្លៃជាប្រភេទ char ក្នុងជំពូកមុនក៏ដោយ យើងនៅខ្វះមធ្យោបាយងាយស្រួលសម្រាប់ដំណើរការស៊េរីតួអក្សរ (ហៅថា string ក្នុងភាសា C)។ ក្នុងជំពូកនេះ យើងនឹងលើកមកសិក្សាទាំង string តម្លៃថេរ (literals) និងអញ្ញាត string ដែលអាចប្តូរបានក្នុងប្រតិបត្តិនៃកម្មវិធី។

១. **អ្វីទៅ** String ?

String គឺជាបណ្តុំតួអក្សរឬសំណុំតួអក្សរ ដែលគេប្រើ array ប្រភេទ char នៅក្នុងភាសា C។ string គឺជា array មួយវិមាត្រប្រភេទ char ដែលបញ្ចប់ដោយ null character '\0'។ ម៉្យាងវិញទៀត string គឺជាការបណ្តុំតួអក្សរដែលបិទអមដោយសញ្ញា " " ។

កំណត់សម្គាល់៖

String ជានិច្ចជាកាលសរសេរដាក់ក្នុងសញ្ញា " " ។ រីឯតួអក្សរសរសេរដាក់ក្នុងសញ្ញា ' ' ។

២. **ការប្រកាស** String

String ត្រូវបានប្រកាសក្នុងភាសា C ស្រដៀងទៅនឹង arrays ដែរ។ វាខុសគ្នាត្រង់ String គឺជាប្រភេទ char ។

ឧទាហរណ៍ ៖ char s[5];

s[0]	s[1]	s[2]	s[3]	s[4]

៣. **ការកំណត់តម្លៃដំបូងឲ្យ** array string

String ត្រូវបានកំណត់តម្លៃដំបូងតាមវិធីផ្សេងៗគ្នាក្នុងភាសា C។

ឧទាហរណ៍ ៖

```
char str[ ] = "abcd";  ឬ
char str[5] = "abcd";  ឬ
```

```
char str[5] = {'a', 'b', 'c', 'd', '\0'};   ឬ
char str[ ] = {'a', 'b', 'c', 'd', '\0'};
```

នៅក្នុងភាសា C, String អាចកំណត់តម្លៃដំបូងដោយប្រើ pointer ផងដែរ។

ឧទាហរណ៍ ៖

```
char  *ch = "abcd";
```

s[0]	s[1]	s[2]	s[3]	s[4]
a	b	c	d	\0

៤. **របៀបអានបញ្ចូល** string **ពី** keyboard

កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើ string ដែលគេបានបញ្ចូលពី keyboard ។

```
#include <stdio.h>
#include <conio.h>
void main(){
   char name[10];
   printf("Enter name: ");
   scanf("%s", name);
   printf("Your name is: %s.", name);
   getch();
}
```

កំណត់សម្គាល់ ៖

អញ្ញាត string អាចទទួលយកតែមួយពាក្យ។ ពីព្រោះវាជួបនឹងដំណកឃ្លា នោះអនុគមន៍ scanf() ត្រូវបញ្ចប់។ ដើម្បីបញ្ចៀសបញ្ហានេះ គេអាចប្រើអនុគមន៍ gets()។

ទម្រង់ទូទៅរបស់វា គឺ៖

```
char str[5];
```

gets(str);

- អនុគមន៍ gets() ត្រូវបានប្រើសម្រាប់ទទួលយកតម្លៃជា string ពី keyboard។ គេប្រើ gets() ដែលអាចទទួលយកការបញ្ចូលពាក្យលើសពី មួយពាក្យ។
- អនុគមន៍ puts() ត្រូវបានប្រើសម្រាប់បញ្ចេញព័តមានមកលើអេក្រង់។ ជាទូទៅអនុគមន៍ puts() ត្រូវបានប្រើជាមួយនឹងអនុគមន៍ gets()។

ចូរសង្កេតមើលឧទាហរណ៍ខាងក្រោមបង្ហាញពីការប្រើ string ជាមួយនឹងអនុគមន៍ gets() និង អនុគមន៍ puts()។

```
#include <stdio.h>
#include <conio.h>
void main() {
   char str[30];
   printf("Enter any string: ");
   gets(str);
   printf("String are: ");
   puts(str);
   getch();
}
```

កំណត់ចំណាំ ៖

យើងត្រូវយកចិត្តទុកដាក់ចំពោះការប្រកាស string មួយ និងការកំណត់តម្លៃដំបូងទៅឲ្យ string នោះ។

- ការប្រកាសទំហំ string តម្រូវការចាំបាច់ដោយបញ្ជាក់ទំហំត្រឹមត្រូវ បើមិនដូច្នោះទេ វានឹងនាំឲ្យ error កើតឡើង។

  ឧទាហរណ៍ ៖ char str[ ]; // មិនត្រឹមត្រូវទេ
  char str[10]; // ត្រឹមត្រូវ
- ការប្រកាសទំហំ string ត្រូវតែជាចំនួនគត់វិជ្ជមាន មិនអាចអវិជ្ជមានឬ សូន្យឡើយ ពោលគឺធំជាងសូន្យ។

ឧទាហរណ៍ ៖ char str[ ]; // មិនត្រឹមត្រូវទេ

char str[-1]; // មិនត្រឹមត្រូវទេ

char str[0]; // មិនត្រឹមត្រូវទេ

- ចំពោះ string ដែលគេអាចប្រើបានតាមរយៈ pointer នោះ មិនអាចយក pointer ទៅកំណត់តម្លៃជាមួយ char បានឡើយ។

ឧទាហរណ៍ ៖ char *p = "abc"; // ត្រឹមត្រូវ

*p = 'd'; // មិនត្រឹមត្រូវទេ

- ការកំណត់តម្លៃដំបូងឲ្យ string បើចំនួនតួអក្សរដែលត្រូវកំណត់ឲ្យ string មិនបានពេញលេញ នោះ វានឹងត្រូវបំពេញដោយ NULL។

ឧទាហរណ៍ ៖ char str[5] = {'5', '+', 'A'};

str[0] ----> 5

str[1] ----> +

str[2] ----> A

str[3] ----> NULL

str[4] ----> NULL

៥. **ការប្រើ** String **ដែលមានក្នុង** Library

៥.១ **ការប្រើអនុគមន៍** strcpy( )

អនុគមន៍ strcpy() ប្រើសម្រាប់ចម្លងតម្លៃរបស់ string មួយទៅឲ្យ string មួយទៀត។ ទម្រង់ទូទៅរបស់វា គឺ៖

```
char * strcpy ( char * destination, const char * source );
```

ឧទាហរណ៍ ៖

strcpy(str1, str2) ; បានន័យថាវាចម្លងតម្លៃ str2 ទៅឲ្យ str1។

strcpy(str2, str1) ; បានន័យថាវាចម្លងតម្លៃ str1 ទៅឲ្យ str2។

ប្រសិនបើប្រវែង destination string ខ្លីជាង source string នោះតម្លៃរបស់ source string ទាំងមូលមិនត្រូវបានចម្លងទៅឲ្យ destination string ឡើយ។ ឧទាហរណ៍ destination string មានប្រវែង

20 តួ ហើយ source string មានប្រវែង 30 តួ។ ពេលនោះ មានតែ 20 តួអក្សររបស់ source string ត្រូវបានចម្លងទៅឲ្យ destination string ហើយនៅសល់ 10 តួអក្សរ វាមិនធ្វើការចម្លងឡើយ ពោលគឺវាតម្រឹមចោល។

កម្មវិធីខាងក្រោមបង្ហាញពីការចម្លងតម្លៃរបស់ source string "C Programming" ទៅឲ្យ target string ដោយប្រើអនុគមន៍ strcpy()។

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main(){
   char source[ ] = "C Programming" ;
   char target[20]= "" ;
   printf ( "\nsource string = %s", source);
   printf ( "\ntarget string = %s", target);
   strcpy ( target, source );
   printf ( "\ntarget string after strcpy() = %s", target);
   getch();
}
```

៥.២ ការប្រើអនុគមន៍ strncpy( )

អនុគមន៍ strncpy() ត្រូវបានប្រើសម្រាប់ចម្លងផ្នែកមួយនៃតម្លៃរបស់ string មួយទៅឲ្យ string មួយទៀត។ ទម្រង់ទូទៅរបស់វា គឺ៖

```
char * strncpy ( char * destination, const char * source, size_t num );
```

<u>ឧទាហរណ៍</u> ៖

strncpy(str1, str2, 4) ; បានន័យថាវាចម្លងតម្លៃ 4 តួអក្សរដំបូងរបស់ str2 ទៅឲ្យ str1។

strncpy(str2, str1, 4) ; បានន័យថាវាចម្លងតម្លៃ 4 តួអក្សរដំបូងរបស់ str1 ទៅឲ្យ str2។

បើសិនជាប្រវែង destination string ខ្លីជាងប្រវែង source string នោះ តម្លៃរបស់ source string ទាំងមូលនឹងមិនត្រូវចម្លងទៅឲ្យ destination string ឡើយ។ ឧទាហរណ៍ destination string មានប្រវែង 20 តួអក្សរ ហើយ source string មានប្រវែង 30 តួអក្សរ។ បើសិនយើងចង់ចម្លង 25 តួអក្សរពី source string ដោយប្រើអនុគមន៍ strncpy() នោះមានតែ 20 តួអក្សររបស់ source string ធ្វើការចម្លងទៅឲ្យ destination string ហើយនៅសល់ 5 តួអក្សរដែលមិនធ្វើការចម្លង រួចវាតម្រឹមចោល។

កម្មវិធីខាងក្រោម មានតែ 5 តួអក្សររបស់ source string "C programming" ដែលធ្វើការចម្លងទៅឲ្យ target string ដោយប្រើអនុគមន៍ strncpy()។

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main(){
   char source[] = "C Programming" ;
   char target[20]= "" ;
   printf("\nsource string = %s", source);
   printf("\ntarget string = %s", target);
   strncpy(target, source, 5) ;
   printf("\ntarget string after strncpy() = %s", target);
   getch();
}
```

៥.៣ ការប្រើអនុគមន៍ strlen( )

អនុគមន៍ strlen() របស់ភាសា C ប្រើសម្រាប់ដឹងបាននូវប្រវែង string ដែលគេឲ្យ។ ទម្រង់ទូទៅរបស់វា គឺ៖

size_t strlen (const char * str);

អនុគមន៍ strlen() រាប់ចំនួនតួអក្សរនៅក្នុង string ដែលគេឲ្យ ហើយតម្លៃជាចំនួនគត់ integer។ វាឈប់រាប់តួអក្សរនៅពេលជួបតួអក្សរ null ពីព្រោះ តួអក្សរ null បញ្ជាក់ប្រាប់នូវទីបញ្ចប់នៃ string។

កម្មវិធីខាងក្រោមជាឧទាហរណ៍មួយបង្ហាញប្រវែងតួអក្សរនៅក្នុង string "Computer Science Department" ដោយប្រើអនុគមន៍ strlen()។ លទ្ធផលនៃកម្មវិធីនេះគឺទទួលបានប្រវែង 27 តួអក្សរ។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
   int len;
   char array[30]= "Computer Science Department";
   len = strlen(array);
   printf("\string length  = %d \n", len);
   getch();
}
```

៥.៤ ការប្រើអនុគមន៍ strcat( )

អនុគមន៍ strcat() នៅក្នុងភាសា C តភ្ជាប់ string ពីរដែលគេឲ្យ។ វាតភ្ជាប់ source string នៅខាងចុង destination string។ ទម្រង់ទូទៅរបស់វា គឺ៖

```
char * strcat ( char * destination, const char * source );
```

<u>ឧទាហរណ៍</u> ៖

strcat(str2, str1) ; បានន័យថា str1 តភ្ជាប់ទៅខាងចុង str2។

strcat(str1, str2) ; បានន័យថា str2 តភ្ជាប់ទៅខាងចុង str1។

ដូចយើងបានដឹងហើយថា string និមួយៗនៅក្នុងភាសា C ត្រូវបញ្ចប់ដោយតួអក្សរ null ('\0')។ នៅក្នុងការប្រើអនុគមន៍ strcat() តួអក្សរ null របស់ destination string ត្រូវបានសរសេរជាន់ដោយ តួ

អក្សរទីមួយរបស់ source string ហើយតួអក្សរ null ត្រូវបានបន្ថែមនៅខាងចុងនៃ destination string ថ្មី ដែលបង្កើតឡើងបន្ទាប់ពីប្រើអនុគមន៍ strcat()។

កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើអនុគមន៍ strcat() ដែលតភ្ជាប់ string ពីរ គឺ "We are learning" និង " C programming." ។

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main(){
   char source[] = " C programming.";
   char target[] = "We are learning";
   printf ( "\nSource string = %s", source);
   printf ( "\nTarget string = %s", target);
   strcat ( target, source ) ;
   printf ( "\nTarget string after strcat() = %s", target);
   getch();
}
```

៥.៥ ការប្រើអនុគមន៍ strncat( )

អនុគមន៍ strncat() នៅក្នុងភាសា C ត្រូវបានប្រើសម្រាប់តភ្ជាប់ (បន្ថែម) ផ្នែកមួយនៃ string ទៅផ្នែកខាងចុងនៃ string ផ្សេងទៀត។ ទម្រង់ទូទៅរបស់វា គឺ៖

```
char * strncat ( char * destination, const char * source, size_t num );
```

<u>ឧទាហរណ៍</u> ៖

strncat(str2, str1, 3) ; បានន័យថា 3 តួអក្សរខាងដើមនៃ str1 តភ្ជាប់ទៅខាងចុង str2។

strncat(str1, str2, 3) ; បានន័យថា 3 តួអក្សរខាងដើមនៃ str2 តភ្ជាប់ទៅខាងចុង str1។

ដូចយើងបានដឹងហើយថា string និមួយៗនៅក្នុងភាសា C ត្រូវបញ្ចប់ដោយតួអក្សរ null ('\0')។ នៅក្នុងការប្រើអនុគមន៍ strncat() តួអក្សរ null របស់ destination string ត្រូវបានសរសេរជាន់ដោយតួអក្សរទីមួយរបស់ source string ហើយតួអក្សរ null ត្រូវបានបន្ថែមនៅខាងចុងនៃ destination string ថ្មីដែលបង្កើតឡើងបន្ទាប់ពីប្រើអនុគមន៍ strncat()។

កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើអនុគមន៍ strncat() ដែលតភ្ជាប់ 7 តួអក្សរដំបូងនៃ string " C programming." នៅខាងចុងនៃ string "We are learning" ដោយប្រើអនុគមន៍ strncat() ហើយលទ្ធផលទទួលបានគឺ "We are learning C prog"។

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main(){
   char source[] = " C programming.";
   char target[] = "We are learning";
   printf ( "\nSource string = %s", source);
   printf ( "\nTarget string = %s", target);
   strncat ( target, source, 7) ;
   printf ( "\nTarget string after strcat() = %s", target);
   getch();
}
```

៥.៦ ការប្រើអនុគមន៍ strcmp( )

អនុគមន៍ strcmp() នៅក្នុងភាសា C ប្រើសម្រាប់ធ្វើការប្រៀបធៀប string ពីរដែលគេឲ្យ រួចហើយឲ្យតម្លៃសូន្យបើសិនជា string ទាំងពីរនេះដូចគ្នា។ បើប្រវែង string1 តូចជាង string2 នោះវានឹងតម្លៃតូចជាងសូន្យ។ បើប្រវែង string1 ធំជាង string2 នោះវានឹងតម្លៃធំជាងសូន្យ។

ទម្រង់ទូទៅរបស់វា គឺ៖

```
int strcmp ( const char * str1, const char * str2 );
```

អនុគមន៍ strcmp() មានលក្ខណៈប្រកាន់តួអក្សតូចធំណាស់ បានន័យថា "A" និង "a" ចាត់ទុកតួអក្សរពីរផ្សេងគ្នា។

កម្មវិធីខាងក្រោមបង្ហាញពីការប្រៀបធៀប string ពីរដោយប្រើអនុគមន៍ strcmp()។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
   char str1[] = "RUPP" ;
   char str2[] = "DCS" ;
   int i, j, k ;
   i = strcmp(str1, "Programming" ) ;
   j = strcmp(str1, str2 ) ;
   k = strcmp(str1, "C" ) ;
   printf ( "\n%d %d %d", i, j, k ) ;
   getch();
}
```

៥.៧ ការប្រើអនុគមន៍ strcmpi( )

អនុគមន៍ strcmpi() ដូចគ្នាទៅនឹងអនុគមន៍ strcmp() ដែរ តែអនុគមន៍ strcmpi() មិនប្រកាន់តួអក្សរតូចធំឡើយ បានន័យថា តួអក្សរ "A" និង តួអក្សរ "a" ចាត់ទុកតួអក្សរដូចគ្នា។ ចំណែក អនុគមន៍ strcmp() ចាត់ទុកតួអក្សរ "A" និង តួអក្សរ "a" ខុសគ្នា។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
   char str1[] = "RUPP" ;
   char str2[] = "DCS" ;
   int i, j, k ;
   i = strcmpi(str1, "programming" ) ;
```

```
    j = strcmpi(str1, str2 ) ;
    k = strcmpi(str1, "C" ) ;
    printf ( "\n%d %d %d", i, j, k ) ;
    getch();
}
```

៥.៨ ការប្រើអនុគមន៍ strchr( )

អនុគមន៍ strchr() នៅក្នុងភាសា C ឲ្យតម្លៃជា pointer ចង្អុលទៅកាន់តួអក្សរដែលជួបប្រទះដំបូងគេនៅក្នុង string ដែលគេឲ្យ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

char *strchr(const char *str, int character);

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ strchr() ដើម្បីរកទីតាំងនៃតួអក្សរ 'i' ដែលជួបប្រទះដំបូងនៅក្នុង string "This is a string for testing" ។ តួអក្សរ 'i' ស្ថិតនៅក្នុងទីតាំង 3 ហើយ pointer នឹងឲ្យតម្លៃនៅត្រង់តួអក្សរ 'i' ដែលបានជួបប្រទះដំបូងគេ។

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main(){
  char string[55] = "This is a string for testing";
  char *p;
  p = strchr(string, 'i');
  printf("Character i is found at position %d\n",
          p-string+1);
  printf("First occurrence of character \"i\" in \"%s\" is"
        \ " \"%s\"", string, p);
  getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

Character i is found at position 3

First occurrence of character "i" in "This is a string for testing" is "is is a string for testing"

បើសិនជាយើងចង់រកតួអក្សរនិមួយៗដែលបានជួបប្រទះក្នុង string ដែលគេឲ្យនោះ គេអាចប្រើកម្មវិធីខាងក្រោម៖

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
  char string[55] ="This is a string for testing";
  char *p;
  int k = 1;
  p = strchr(string,'i');
  while(p!=NULL){
    printf("Character i found at position %d\n",p-string+1);
    printf("Occurrence of character \"i\" : %d \n",k);
    printf("Occurrence of character \"i\" in \"%s\" is \"%s" \
           "\"\n",string, p);
    p=strchr(p+1,'i');
    k++;
  }
  getch();
}
```

៥.៩ ការប្រើអនុគមន៍ strrchr( )

អនុគមន៍ strrchr() នៅក្នុងភាសា C ឲ្យតម្លៃជា pointer ចង្អុលទៅកាន់តួអក្សរដែលជួបប្រទះនៅខាងចុងនៃ string ដែលគេឲ្យ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

char *strrchr(const char *str, int character);

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ strrchr() ដើម្បីរកទីតាំងនៃតួអក្សរ 'i' ដែលជួបប្រទះផ្នែកខាងចុងមានក្នុង string "This is a string for testing" ។ តួអក្សរ 'i' ស្ថិតនៅក្នុងទីតាំង 26 ហើយ pointer នឹងឲ្យតម្លៃនៅត្រង់តួអក្សរ 'i' ដែលបានជួបប្រទះនៅចុងក្រោយគេ។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
  char string[55] ="This is a string for testing";
  char *p;
  p = strrchr(string,'i');
  printf ("Character i is found at position %d\n",
         p-string+1);

  printf("Last occurrence of character \"i\" in \"%s\" is"
        \ " \"%s\"", string, p);
  getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

Character i is found at position 26

Last occurrence of character ″i″ in ″This is a string for testing″ is ″ing″

៥.១០ ការប្រើអនុគមន៍ strstr( )

អនុគមន៍ strstr() នៅក្នុងភាសា C ឲ្យតម្លៃជា pointer ចង្អុលទៅកាន់ string ដែលជួបប្រទះដំបូងគេនៅក្នុង string ដែលគេឲ្យ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

char *strstr(const char *str1, const char *str2);

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ strstr() ដើម្បីរកទីតាំងនៃ string ដែលជួបប្រទះដំបូងនៅក្នុង string "This is a test string for testing" ។ pointer នឹងឲ្យតម្លៃនៅត្រង់ string "test" ដែលបានជួបប្រទះដំបូងគេ។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
  char string[55] = "This is a test string for testing";
  char *p;
  p = strstr(string, "test");
  if (p){
    printf("string found\n" );
    printf("First occurrence of string \"test\" in \"%s\"
          is"\ " \"%s\"",string, p);
  }
  else
    printf("string not found\n" );
  getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

string found

First occurrence of string "test" in "This is a test string for testing" is

"test string for testing"

៥.១១ ការប្រើអនុគមន៍ strrstr( )

អនុគមន៍ strrstr() នៅក្នុងភាសា C ឲ្យតម្លៃជា pointer ចង្អុលទៅកាន់ string ដែលជួបប្រទះខាងក្រោយគេនៅក្នុង string ដែលគេឲ្យ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

char *strrstr(const char *str1, const char *str2);

អនុគមន៍នេះគឺជាអនុគមន៍មិនមានលក្ខណៈស្តង់ដារដែលអាចមិនមានក្នុង library ស្តង់ដារនៃភាសា C។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ strrstr() ដើម្បីរកទីតាំងនៃ string ដែលជួបប្រទះនៅផ្នែកខាងចុងនៃ string "test" ក្នុង string "This is a test string for testing" ។ pointer នឹងឲ្យតម្លៃនៅត្រង់ string "test" ដែលបានជួបប្រទះមុនគេផ្នែកខាងចុង។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
  char string[55] = "This is a test string for testing";
  char *p;
  p = strrstr(string, "test");
  if (p){
    printf("string found\n" );
    printf ("Last occurrence of string \"test\" in \"%s\"
            is"\ " \"%s\"", string, p);
  } else
    printf("string not found\n" );
  getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

string found

Last occurrence of string "test" in "This is a test string for testing" is "testing"

៥.១២ ការប្រើអនុគមន៍ strdup( )

អនុគមន៍ strdup() របស់ភាសា C ប្រើសម្រាប់ចម្លង string ដែលគេឲ្យ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

```
char *strdup(const char *string);
```

អនុគមន៍នេះគឺជាអនុគមន៍មិនមានលក្ខណៈស្តង់ដារដែលអាចមិនមានក្នុង library ស្តង់ដារនៃភាសា C។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ strdup() ដើម្បីចម្លង string "RUPP" រួចបង្ហាញលទ្ធផលមកក្រៅ។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
   char *p1 = "RUPP";
   char *p2;
   p2 = strdup(p1);
   printf("Duplicated string is : %s", p2);
   getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

```
Duplicated string is : RUPP
```

៥.១៣ ការប្រើអនុគមន៍ strlwr( )

អនុគមន៍ strlwr() ប្រើសម្រាប់បំលែង string ដែលគេឲ្យទៅជាអក្សរតូចទាំងអស់។
ទម្រង់ទូទៅរបស់វា គឺ ៖

```
char *strlwr(char *string);
```

អនុគមន៍នេះគឺជាអនុគមន៍មិនមានលក្ខណៈស្តង់ដារដែលអាចមិនមានក្នុង library ស្តង់ដារនៃភាសា C។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះ string "MODIFY This String To LOwer" ត្រូវបានបំលែងទៅជាអក្សរតូចទាំងអស់ដោយប្រើអនុគមន៍ strlwr() រួចហើយបង្ហាញលទ្ធផលមកក្រៅ គឺ "modify this string to lower"។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
   char str[] = "MODIFY This String To LOwer";
   printf("%s\n", strlwr (str));
   getch();
}
```

៥.១៤ ការប្រើអនុគមន៍ strupr( )

អនុគមន៍ strupr() ប្រើសម្រាប់បំលែង string ដែលគេឲ្យទៅជាអក្សរធំទាំងអស់។ ទម្រង់ទូទៅរបស់វា គឺ ៖

```
char *strupr(char *string);
```

អនុគមន៍នេះគឺជាអនុគមន៍មិនមានលក្ខណៈស្តង់ដារដែលអាចមិនមានក្នុង library ស្តង់ដារនៃភាសា C។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះ string "MODIFY This String To Upper" ត្រូវបានបំលែងទៅជាអក្សរតូចទាំងអស់ដោយប្រើអនុគមន៍ strlwr() រួចហើយបង្ហាញលទ្ធផលមកក្រៅ គឺ "MODIFY THIS STRING TO UPPER"។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
   char str[ ] = "Modify This String To Upper";
   printf("%s\n", strupr(str));
   getch();
```

```
}
```

៥.១៥ ការប្រើអនុគមន៍ strrev( )

អនុគមន៍ strrev() នៅក្នុងភាសា C ប្រើសម្រាប់ធ្វើចម្រាស់ string ដែលគេឲ្យ។ ទម្រង់ទូទៅរបស់វា គឺ ៖

```
char *strrev(char *string);
```

អនុគមន៍នេះគឺជាអនុគមន៍មិនមានលក្ខណៈស្តង់ដារដែលអាចមិនមានក្នុង library ស្តង់ដារនៃភាសា C។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះ string "Hello" ត្រូវធ្វើចម្រាស់ដោយប្រើអនុគមន៍ strrev() រួចបង្ហាញលទ្ធផលមកក្រៅ គឺ "olleH"។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
   char name[30] = "Hello";
   printf("String before strrev() : %s\n", name);
   printf("String after strrev()  : %s", strrev(name));
   getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

```
String before strrev( ) : Hello
String after strrev( )     : olleH
```

៥.១៦ ការប្រើអនុគមន៍ strset( )

អនុគមន៍ strset() នៅក្នុងភាសា C ប្រើសម្រាប់កំណត់តួអក្សរទាំងអស់នៅក្នុង string ទៅជាតួអក្សរដែលគេឲ្យ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

char *strset(char *string, int c);

អនុគមន៍នេះគឺជាអនុគមន៍មិនមានលក្ខណៈស្តង់ដារដែលអាចមិនមានក្នុង library ស្តង់ដារនៃភាសា C។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះគ្រប់តួអក្សរទាំងអស់នៃ string "Test String" ត្រូវដាក់ទៅជា "#" ដោយប្រើអនុគមន៍ strset() រួចហើយបង្ហាញលទ្ធផលមកក្រៅ គឺ "##########"។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
   char str[20] = "Test String";
   printf("Original string is : %s\n", str);
   printf("Test string after strset() : %s\n",
           strset(str,'#'));
   printf("After string set: %s\n", str);
   getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

Original string is : Test String

Test string after strset() : ##########

After string set : ##########

៥.១៧ ការប្រើអនុគមន៍ strnset( )

អនុគមន៍ strnset() នៅក្នុងភាសា C ប្រើសម្រាប់កំណត់តួអក្សររបស់ផ្នែកមួយនៃ string ទៅជាតួអក្សរដែលគេឲ្យ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

char *strnset(char *string, int c);

អនុគមន៍នេះគឺជាអនុគមន៍មិនមានលក្ខណៈស្តង់ដារដែលអាចមិនមានក្នុង library ស្តង់ដារនៃភាសា C។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះចំនួន 4 តួអក្សរនៃ string "Test String" ត្រូវបានដាក់ទៅជា "#" ដោយប្រើអនុគមន៍ strnset() រួចហើយបង្ហាញលទ្ធផលមកក្រៅ គឺ "#### String"។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
   char str[20] = "Test String";
   printf("Original string is : %s\n", str);
   printf("Test string after string n set" \
          " : %s\n", strnset(str,'#',4));
   printf("After string n set : %s\n", str);
   getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

```
Original string is : Test String
Test string after string set : #### String
After string n set : #### String
```

៥.១៨ ការប្រើអនុគមន៍ strtok( )

អនុគមន៍ strtok() នៅក្នុងភាសា C ប្រើសម្រាប់ធ្វើការញែក string ដែលគេឲ្យដោយប្រើ delimiter។

ទម្រង់ទូទៅរបស់វា គឺ ៖

```
char * strtok ( char * str, const char * delimiters );
```

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះគេបញ្ចូល string "Test,string1,Test,string2:Test:string3" ត្រូវបានធ្វើការញែកដោយប្រើអនុគមន៍ strtok()។ delimiter សញ្ញាក្បៀស ", " ប្រើសម្រាប់ញែក string តូចៗចេញពី string ដែលបានបញ្ចូល។

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
void main(){
   char string[50] = "Test,string1,Test,string2:Test:string3";
   char *p;
   printf ("String \"%s\" is split into tokens:\n", string);
   p = strtok(string, ",:");
   while (p!= NULL){
     printf ("%s\n", p);
     p = strtok(NULL, ",:");
   }
   getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

```
String "Test,string1,Test,string2:Test:string3" is split into tokens:
Test
string1
Test
string2
Test
string3
```

៥.១៩ ការប្រើអនុគមន៍ stricmp( )

អនុគមន៍ stricmp() នេះប្រើសម្រាប់ធ្វើការប្រៀបធៀប string ពីរ ដោយមិនប្រកាន់តួអក្សរតូចធំឡើយ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

int stricmp(const char *s1, const char *s2)

វានឹងឲ្យតម្លៃជាចំនួនគត់ int ធំជាងសូន្យ ស្មើសូន្យឬ តូចជាងសូន្យ អាស្រ័យដោយ s1 ធំជាង ស្មើ ឬ តូចជាង s2។ គេត្រូវបញ្ចូល header file ឈ្មោះ "string.h"។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ stricmp()។

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main(){
   char *str1="HELLO";
   char *str2="Hello";
   int i;
   printf("str1 : %s\n", str1);
   printf("str2 : %s\n", str2);
   i = stricmp(str1, str2);
   if (i==0)
      printf("str1 and str2 are identical\n");
   else if (i < 0)
      printf("str1 < str2\n");
   else
      printf("str1 > str2\n");
    getch();
}
```

៥.២០ ការប្រើអនុគមន៍ strnicmp( )

អនុគមន៍ strnicmp() នេះប្រើសម្រាប់ធ្វើការប្រៀបធៀបផ្នែកមួយនៃ string (តាមចំនួនកំណត់) ទៅនឹង string មួយទៀតដោយមិនប្រកាន់តួអក្សរតូចធំឡើយ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

int strnicmp(const char *s1, const char *s2, size_t n)

វានឹងឲ្យតម្លៃជាចំនួនគត់ int ធំជាងសូន្យ ស្មើសូន្យឬ តូចជាងសូន្យ អាស្រ័យដោយ s1 ធំជាង ស្មើ ឬ តូចជាង s2។ គេត្រូវបញ្ចូល header file ឈ្មោះ "string.h"។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ strnicmp() ដោយធ្វើការប្រៀបធៀបចំនួន 5 តួអក្សរនៃ string "HelloWorld" ទៅនឹង string "HELLO"។

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main(){
   char *str1="HELLO";
   char *str2="HelloWorld";
   int i;
   printf("str1 : %s\n", str1);
   printf("str2 : %s\n", str2);
   i = strnicmp(str1, str2, 5);
   if (i==0)
      printf("str1 and str2 are identical\n");
   else if (i < 0)
      printf("str1 < str2\n");
   else
      printf("str1 > str2\n");
    getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

str1 : HELLO

str2 : HelloWorld

str1 and str2 are identical

៥.២១ ការប្រើអនុគមន៍ atoi( )

អនុគមន៍ atoi() នៅក្នុងភាសា C ប្រើសម្រាប់បំលែងប្រភេទទិន្នន័យ string ទៅជាប្រភេទទិន្នន័យ int ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

```
int atoi (const char * str);
```

ដើម្បីប្រើអនុគមន៍នេះបាន យើងត្រូវបញ្ចូល header file "stdlib.h"។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ atoi()។

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void main(){
   char a[10] = "100";
   int value = atoi(a);
   printf("Value = %d\n", value);
   getch();
}
```

៥.២២ ការប្រើអនុគមន៍ atof( )

អនុគមន៍ atof() នៅក្នុងភាសា C ប្រើសម្រាប់បំលែងប្រភេទទិន្នន័យ string ទៅជាប្រភេទទិន្នន័យ float ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

```
int atof (const char * str);
```

ដើម្បីប្រើអនុគមន៍នេះបាន យើងត្រូវបញ្ចូល header file "stdlib.h"។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ atof()។

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void main(){
   char a[10] = "3.14";
   float pi = atof(a);
   printf("Value of pi = %f\n", pi);
   getch();
}
```

៥.២៣ ការប្រើអនុគមន៍ atol( )

អនុគមន៍ atol() នៅក្នុងភាសា C ប្រើសម្រាប់បំលែងប្រភេទទិន្នន័យ string ទៅជាប្រភេទទិន្នន័យ long ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

```
int atol (const char * str);
```

ដើម្បីប្រើអនុគមន៍នេះបាន យើងត្រូវបញ្ចូល header file "stdlib.h"។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ atol()។

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void main(){
   char a[20] = "1000000000";
   long value = atol(a);
   printf("Value = %ld\n", value);
   getch();
}
```

៥.២៤ ការប្រើអនុគមន៍ itoa( )

អនុគមន៍ itoa() នៅក្នុងភាសា C ប្រើសម្រាប់បំលែងប្រភេទទិន្នន័យ int ទៅជាប្រភេទទិន្នន័យ string ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

```
int itoa (int value, char * str, int base);
```

ដើម្បីប្រើអនុគមន៍នេះបាន យើងត្រូវបញ្ចូល header file "stdlib.h"។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ itoa()។

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void main(){
    int a=54325;
    char buffer[20];
    itoa(a, buffer,2);   // here 2 means binary
    printf("Binary value = %s\n", buffer);

    itoa(a, buffer,10);   // here 10 means decimal
    printf("Decimal value = %s\n", buffer);

    itoa(a, buffer,16);   // here 16 means Hexadecimal
    printf("Hexadecimal value = %s\n", buffer);
    getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

```
Binary value = 1101010000110101
Decimal value = 54325
Hexadecimal value = D435
```

៥.២៥ ការប្រើអនុគមន៍ ltoa( )

អនុគមន៍ ltoa() នៅក្នុងភាសា C ប្រើសម្រាប់បំលែងប្រភេទទិន្នន័យ long ទៅជាប្រភេទទិន្នន័យ string ។

ទម្រង់ទូទៅរបស់វា គឺ ៖

```
int ltoa (long N, char * str, int base);
```

ដើម្បីប្រើអនុគមន៍នេះបាន យើងត្រូវបញ្ចូល header file "stdlib.h"។

<u>ឧទាហរណ៍</u> ៖ កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ ltoa()។

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void main(){
    long a=10000000;
    char buffer[50];
    ltoa(a, buffer,2);   // here 2 means binary
    printf("Binary value = %s\n", buffer);

    ltoa(a, buffer,10);   // here 10 means decimal
    printf("Decimal value = %s\n", buffer);

    ltoa(a, buffer,16);   // here 16 means Hexadecimal
    printf("Hexadecimal value = %s\n", buffer);
    getch();
}
```

លទ្ធផលទទួលបាន គឺ ៖

```
Binary value = 10011000100101101010000000
Decimal value = 10000000
Hexadecimal value = 989680
```