ជំពូកទី ១០ អំពី File I/O

១. សេចក្តីផ្តើម

នៅពេលដែលគេនិយាយថា Input នោះ មានន័យថា បញ្ចូលទិន្នន័យទៅឲ្យកម្មវិធីមួយ។ ការបញ្ចូលទិន្នន័យអាចឲ្យក្រោមទម្រង់នៃ file មួយឬ តាមខ្សែបន្ទាត់សម្រាប់បញ្ចូលពាក្យបញ្ជា។ ភាសាសរសេរកម្មវិធី C ផ្តល់នូវសំណុំអនុគមន៍ដែលមានស្រាប់សម្រាប់អានបញ្ចូលទិន្នន័យដែលគេឲ្យហើយបញ្ចូលវាទៅក្នុងកម្មវិធីតាមតម្រូវការ។

នៅពេលដែលគេនិយាយ Output នោះ មានន័យថា បង្ហាញទិន្នន័យនៅលើអេក្រង់ នៅឯ printer ឬក្រោមទម្រង់ជា file មួយ។ ភាសា សរសេរកម្មវិធី C ផ្តល់នូវសំណុំអនុគមន៍ដែលមានស្រាប់សម្រាប់បញ្ចេញទិន្នន័យនៅលើអេក្រង់កុំព្យូទ័រក៏ដូចនឹងរក្សាទុកទិន្នន័យនៅក្នុង file ក្រោមទម្រង់ជា binary ឬ text file។

២. អំពី standard files

ការសរសេរកម្មវិធីភាសា C ចាត់ទុកឧបករណ៍ទាំងអស់ដូចទៅនឹង files។ ហេតុនេះ ឧបករណ៍ដូចជា អេក្រង់ត្រូវបានសំដែងដូចគ្នាទៅនឹង files ហើយ files ចំនួនបីនៅខាងក្រោម ត្រូវបានបើកដោយស្វ័យប្រវត្តិនៅពេលកម្មវិធីមួយប្រតិបត្តិ ដើម្បីផ្តល់នូវការចូលប្រើ keyboard និងអេក្រង់។

Standard File	File Pointer	Device
Standard input	stdin	Keyboard
Standard output	stdout	Screen
Standard error	stderr	Your screen

File pointer គឺជាមធ្យោបាយសម្រាប់ចូលប្រើ file ក្នុងគោលបំណងអានឬសរសេរ។ នៅក្នុងផ្នែកនេះ យើងនឹងបង្ហាញពីរបៀបចូលអានតម្លៃពីអេក្រង់ និងរបៀបបង្ហាញលទ្ធផលនៅលើអេក្រង់។

៣. អនុគមន៍ getchar( ) និង putchar( )

អនុគមន៍ int getchar(void) ប្រើសម្រាប់អានតួអក្សរដែលមាននៅបន្ទាប់ពីអេក្រង់ និងឲ្យលទ្ធផលជាចំនួនគត់ integer។ អនុគមន៍នេះ អានម្តងមួយតួអក្សរប៉ុណ្ណោះ។ គេអាចប្រើវិធីនេះ នៅក្នុង loop ករណីគេចង់អានច្រើនតួអក្សរពីអេក្រង់។

អនុគមន៍ int putchar(int c) ប្រើសម្រាប់សរសេរតួអក្សរមកលើអេក្រង់ហើយឲ្យតម្លៃនូវតួអក្សរដូចគ្នា។ អនុគមន៍នេះ សរសេរអក្សរម្តងមួយតួអក្សរប៉ុណ្ណោះ។ គេអាចប្រើវិធីនេះ នៅក្នុង loop ករណីគេចង់សរសេរច្រើនតួអក្សរមកលើអេក្រង់។

ចូរសង្កេតឧទាហរណ៍ទីមួយដូចខាងក្រោមនេះ៖

```
#include <stdio.h>
#include <conio.h>
void main(){
   int c;
   printf( "Enter a value : ");
   c = getchar( );
   printf( "\nYou entered: ");
   putchar( c );
   getch();
}
```

នៅពេលយើងធ្វើការ compiled និង run កម្មវិធីខាងលើនោះ វារង់ចាំឲ្យយើងបញ្ចូលអក្សរមួយចំនួន។ ពេលយើងបញ្ចូលអក្សរមួយឃ្លាហើយ រួចចុច Enter key ពេលនោះកម្មវិធីនឹងដំណើរការដោយអានតែមួយតួអក្សរប៉ុណ្ណោះ និងបង្ហាញតួអក្សរនោះមកលើអេក្រង់ដែលមានលំនាំដូចខាងក្រោម៖

```
Enter a value : this is test ↵
You entered: t
```

ចូរសង្កេតឧទាហរណ៍ទីពីរដូចខាងក្រោមនេះ៖

```
#include <stdio.h>
#include <conio.h>
void main(){
   int c;
   printf("Enter a value : ");
   while ((c=getchar()) != '\0')
      putchar(c);
   getch();
}
```

ចំពោះកម្មវិធីខាងលើនេះ ពេលយើងធ្វើការ compiled និង run កម្មវិធីខាងលើនោះ វារង់ចាំឲ្យយើងបញ្ចូលអក្សរមួយចំនួន។ ពេលយើងបញ្ចូលអក្សរមួយឃ្លាហើយ រួចចុច Enter key ពេលនោះកម្មវិធីនឹងដំណើរការដោយអានតួអក្សរដែលបានបញ្ចូល និងបង្ហាញតួអក្សរទាំងនោះមកលើអេក្រង់ដែលមានលំនាំដូចខាងក្រោម៖

Enter a value : this is test ↵

this is test

៤. អនុគមន៍ gets( ) **និង** puts( )

អនុគមន៍ char *gets(char *s) ប្រើសម្រាប់អានអក្សរមួយជួរពី stdin ទៅដាក់ក្នុង buffer ដែលចង្អុលទៅកាន់ ដោយ s រហូតដល់ជួបតួអក្សរ \n ឬ EOF (End of File)។

អនុគមន៍ int *puts(const char *s) ប្រើសម្រាប់សរសេរ string 's' និង 'a' ដោយមានភ្ជាប់ខាងចុងគឺតួអក្សរ \n ទៅឲ្យ stdout។

ចូរសង្កេតឧទាហរណ៍ខាងក្រោមនេះ៖

```
#include <stdio.h>
#include <conio.h>
void main(){
   char str[100];
   printf("Enter a value : ");
   gets(str);
   printf("\nYou entered: ");
   puts(str);
   getch();
}
```

ពេលយើងធ្វើការ compiled និង run កម្មវិធីខាងលើនោះ វារង់ចាំឲ្យយើងបញ្ចូលអក្សរមួយចំនួន។ ពេលយើងបញ្ចូលអក្សរមួយឃ្លាហើយ រួចចុច Enter key ពេលនោះកម្មវិធីនឹងដំណើរការដោយអានតួអក្សរដែលបានបញ្ចូល និងបង្ហាញតួអក្សរទាំងនោះមកលើអេក្រង់ដែលមានលំនាំដូចខាងក្រោមនេះ ៖

Enter a value : this is test ↵

You entered : this is test

៥. អនុគមន៍ scanf( ) **និង** printf( )

អនុគមន៍ int scanf(const char *format, ...) ប្រើសម្រាប់អានបញ្ចូលទិន្នន័យពី standard input stream stdin ហើយមើលការបញ្ចូលនោះដោយផ្អែកទៅលើទម្រង់ (format) ដែលគេបានផ្តល់ឲ្យ។

អនុគមន៍ int printf(const char *format, ...) ប្រើសម្រាប់សរសេរលទ្ធផលមកលើ standard output stream stdout ហើយឲ្យលទ្ធផលទៅតាមទម្រង់ដែលគេផ្តល់ឲ្យ។

format អាចជា string ថេរ ក៏ប៉ុន្តែអ្នកអាចបញ្ចាក់ %s, %d, %c, %f, -ល- សម្រាប់បង្ហាញឬ អានបញ្ចូល string, integer, character ឬ float តាមរៀងគ្នា។ ចូរយើងពិនិត្យឧទាហរណ៍ខាងក្រោម ដើម្បីយល់បាននូវគំនិតនេះ។

```
#include <stdio.h>
#include <conio.h>
void main(){
   char str[100];
   int i;
   printf( "Enter a value : ");
   scanf("%s %d", str, &i);
   printf( "\nYou entered: %s %d ", str, i);
   getch();
}
```

ពេលយើងធ្វើការ compiled និង run កម្មវិធីខាងលើនោះ វារង់ចាំឲ្យយើងបញ្ចូលតម្លៃមួយចំនួន។ ពេលយើងបញ្ចូលរួចហើយ ចុច Enter key ពេលនោះកម្មវិធីនឹងដំណើរការដោយអានតម្លៃដែលបានបញ្ចូល និងបង្ហាញតម្លៃទាំងនោះមកលើអេក្រង់ដែលមានលំនាំដូចខាងក្រោមនេះ ៖

Enter a value : seven 7

You entered: seven 7

៦. ការប្រើ file

អ្វីទៅហៅថា file ? file គឺជាការប្រមូលផ្តុំ bytes ដែលត្រូវបានផ្ទុកនៅលើឧបករណ៍ផ្ទុកទិន្នន័យ ដូចជា disk។ វាមាន files ពីរប្រភេទនៅក្នុងប្រព័ន្ធមួយ។ នោះគឺ ៖ text files (ASCII) និង binary files។ text files រួមមាន ASCII code នៃលេខ, អក្សរក្រម, និងនិមិត្តសញ្ញា។ binary file មានការប្រមូលផ្តុំនៃ bytes (0 និង 1)។

៧. ការប្រតិបត្តិ file

វាមានការប្រតិបត្តិសំខាន់ៗចំនួន 4 ដែលអាចធ្វើការជាមួយនឹង files នៅក្នុងភាសា C។ ការប្រតិបត្តិសំខាន់ៗ គឺ៖

- ការបើកឬបង្កើត file មួយ
- ការបិទ file
- ការអាន file
- ការសរសេរ file

ចូរយើងសង្កេតទម្រង់ទូទៅនៃការប្រតិបត្តិខាងលើនេះ ៖

៧.១ ការប្រើអនុគមន៍ fopen( )

អនុគមន៍នេះប្រើសម្រាប់បើក file មួយ ដើម្បីធ្វើការប្រតិបត្តិ ដូចជា អាន file, សរសេរ file -ល-។ នៅក្នុងកម្មវិធី C យើងប្រកាស file pointer និងប្រើ fopen() ដូចខាងក្រោម។ អនុគមន៍ fopen() បង្កើត file ថ្មីមួយ បើសិនជា ឈ្មោះ file នោះមិនមាន។

```
FILE *fopen (const char *filename, const char *mode)
```

<u>ឧទាហរណ៍</u> ៖

```
FILE *fp;
fp=fopen ("filename", "mode");
```

ដែលក្នុងនេះ fp គឺជា file pointer ចង្អុលទៅកាន់ "FILE"។ filename គឺជាឈ្មោះ file ដែលមាន path របស់ file ពេញលេញ។ mode សំដៅទៅលើការប្រតិបត្តិដែលត្រូវធ្វើជាមួយ file ។ ឧទាហរណ៍៖ r, w, a, r+, w+ និង a+។

៧.២ ការប្រើអនុគមន៍ fclose( )

អនុគមន៍នេះប្រើសម្រាប់បិទ file ដែលត្រូវបានចង្អុលប្រាប់ដោយ file pointer fp។ នៅក្នុងកម្មវិធីភាសា C, គេអាចបិទ file តាមទម្រង់ដូចខាងក្រោម ៖

int fclose(FILE *fp);

<u>ឧទាហរណ៍</u> ៖

fclose(fp);

៧.៣ ការប្រើអនុគមន៍ fprintf( )

អនុគមន៍នេះប្រើសម្រាប់សរសេរ string ទៅកាន់ file មួយដែលចង្អុលប្រាប់ដោយ fp។ នៅក្នុងកម្មវិធីភាសា C គេសរសេរ string ទៅក្នុង file មួយតាមទម្រង់ខាងក្រោម ៖

int fprintf(FILE *fp, const char *format, …);

<u>ឧទាហរណ៍</u> ៖

fprintf (fp, "some data");

ឬ fprintf (fp, "text %d", variable_name);

កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ fprintf() ដើម្បីសរសេរទិន្នន័យដែលបានទទួលពី keyboard បញ្ចូលទៅក្នុង file មួយ។

```
#include <stdio.h>
#include <conio.h>
void main(){
   FILE *fp;
   int roll;
   char name[25];
   float marks;
   fp = fopen("file2.txt", "w");    // Statement 1
   if(fp == NULL) {
      printf("\nCan't open file or file doesn't exist.");
      getch();
```

```
        return;
    }
    printf("\nEnter Roll : ");
    scanf("%d", &roll);
    printf("\nEnter Name : ");
    scanf("%s", name);
    printf("\nEnter Marks : ");
    scanf("%f", &marks);
    fprintf(fp,"%d %s %f", roll, name, marks);
    printf("\nData written successfully...");
    fclose(fp);
    getch();
}
```

៧.៤ លក្ខណៈ: Mode **នៃការធ្វើប្រតិបត្តិលើ** file **មួយ**

វាមាន mode ជាច្រើននៅក្នុងការបើក file មួយ។ ផ្អែកទៅលើ mode របស់ file, វាអាចបើកសម្រាប់អាន ឬសរសេរ ឬ បន្ថែមអក្សរ។ លក្ខណៈ mode សំខាន់ៗបានរាយឈ្មោះដូចខាងក្រោម៖

- r : បើក text file មួយ (ក្នុងន័យ read mode) និងកំណត់ pointer ទៅកាន់តួអក្សរទីមួយនៅក្នុង file ដែលមានស្រាប់។ វាឲ្យតម្លៃ null បើសិនជា file មិនមាន។
- w : បើក text file មួយ (ក្នុងន័យ write mode)។ វានឹងឲ្យតម្លៃ null បើសិនជា file មិនអាចបើកបាន។ បើវាមិនមានឈ្មោះ file នោះទេ វាបង្កើត file ថ្មីឲ្យ។ តែបើសិនជា file មានហើយ នោះវានឹងសរសេរជាន់ខ្លឹមសារក្នុង file។
- a : បើក text file មួយ (ក្នុងន័យ append mode)។ វានឹងឲ្យតម្លៃ null បើសិនជា file មិនអាចបើកបាន។
- r+ : បើក text file មួយសម្រាប់ read និង write mode ហើយកំណត់ pointer ទៅកាន់តួអក្សរទីមួយនៅក្នុង file។
- w+ : បើក text file មួយសម្រាប់ read និង write mode ហើយកំណត់ pointer ទៅកាន់តួអក្សរទីមួយនៅក្នុង file បើសិនជា file នោះមានហើយ។ ផ្ទុយទៅវិញ វានឹងបង្កើត file ថ្មី។

- a+ : បើក text file មួយសម្រាប់ read និង write mode ហើយកំណត់ pointer ទៅកាន់តួអក្សរទីមួយនៅក្នុង file ករណីវាអាន file ប៉ុន្តែការសរសេរអាចធ្វើឡើងបានត្រឹមតែសរសេរបន្តពីខ្លឹមសារដែលមានក្នុង file នោះ។

<u>កំណត់សម្គាល់:</u> នៅពេលគេចង់ប្រើ binary file នោះ គេត្រូវប្រើ mode ខាងក្រោមជំនួសឲ្យ mode ដែលបានអធិប្បាយខាងលើ ដូចជា ៖

"rb", "wb", "ab", "rb+", "r+b", "wb+", "w+b", "ab+", "a+b"

<u>ឧទាហរណ៍ទី ១</u> ៖

```
// Open, write and close a file
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main(){
    FILE *fp ;
    char data[50];
    // opening an existing file
    printf("Opening the file test.c in write mode.\n");
    fp = fopen("test.c", "w") ;
    if (fp == NULL) {
        printf("Could not open file test.c");
        getch();
        return;
    }
    printf("\nEnter some texts from keyboard" \
           " to write in the file test.c\n") ;
    // getting input from user
    while (strlen(gets(data)) > 0){
       // writing in the file
       fputs(data, fp);
       fputs("\n", fp);
```

```
    }
    // closing the file
    printf("Closing the file test.c");
    fclose(fp) ;
    getch();
}
```

<u>ឧទាហរណ៍ទី</u> ២ ៖ កម្មវិធីខាងក្រោមបង្ហាញពីរបៀបចូលអានខ្លឹមសាររបស់ file ។ ឧបមាថា file មួយឈ្មោះ test.c ដែលមានខ្លឹមសារ "Hi, How are you?" ។ ចូរអានទិន្នន័យនៅក្នុង file នេះដោយសរសេរកម្មវិធីដូចខាងក្រោម ៖

```
// Open, Read and close a file: reading string by string
#include <stdio.h>
#include <conio.h>
void main(){
    FILE *fp;
    char data[50];
    printf("Opening the file test.c in read mode\n");
    fp = fopen("D:/MyData/Cprog/test.c", "r");
    if (fp == NULL){
       printf("Could not open file test.c\n");
       getch();
       return;
    }
    printf("Reading the file test.c : ");
    while(fgets(data, 50, fp)!= NULL)
       printf("%s", data) ;
    printf("\nClosing the file test.c\n");
    fclose(fp) ;
    getch();
}
```

៧.៥ ការប្រើអនុគមន៍ fgetc( )

អនុគមន៍នេះប្រើសម្រាប់អានមួយតួអក្សរពី file មួយ។ វាអានម្តងតែមួយតួអក្សរប៉ុណ្ណោះ ហើយផ្លាស់ទីតាំង file pointer ទៅកាន់ទីតាំងបន្ទាប់ ដើម្បីអានតួអក្សរបន្ទាប់។ ទម្រង់ទូទៅរបស់វា គឺ៖

int fgetc(FILE *fp)

<u>ឧទាហរណ៍</u> ៖

fgetc(fp);

ដែល fp គឺជា file pointer ។

ការប្រើ file នេះនៅក្នុងកម្មវិធី C បង្ហាញពីរបៀបអានខ្លឹមសាររបស់ file មួយ។ ឧបមាថា គេមាន file មួយឈ្មោះ "test.c" ដែលមានខ្លឹមសារ "Hi, How are you?"។ ចូរអានបញ្ចូលទិន្នន័យនេះដោយប្រើអនុគមន៍ fgetc() ។

```
#include <conio.h>
#include <stdio.h>
void main(){
 FILE *fp ;
   char c ;
   printf("Opening the file test.c in read mode.\n");
   fp = fopen("test.c", "r"); // opening an existing file
   if ( fp == NULL ){
     printf("Could not open file test.c\n");
     getch();
     return;
   }
   printf("Reading the file test.c : \n");
   while (1){
     c = (char) fgetc(fp); // reading the file
     if (c == EOF)
        break;
     printf("%c", c);
```

```
    }
    printf("Closing the file test.c\n");
    fclose(fp); // Closing the file
    getch();
}
```

៧.៦ ការប្រើអនុគមន៍ fputc( )

អនុគមន៍នេះត្រូវបានប្រើសម្រាប់សរសេរមួយតួអក្សរទៅក្នុង file មួយ។ វាសរសេរម្តងបានតែមួយតួអក្សរប៉ុណ្ណោះ ហើយផ្លាស់ទីតាំង file pointer ទៅទីតាំងបន្ទាប់ដើម្បីសរសេរតួអក្សរបន្ទាប់ទៀត។ ទម្រង់ទូទៅរបស់វា គឺ ៖

int fputc(int char, FILE *fp)

<u>ឧទាហរណ៍</u> ៖

fputc(ch, fp);

ដែល ch មានតម្លៃជាតួអក្សរ និង fp ជា file pointer ។

កម្មវិធីខាងក្រោមនេះ បង្ហាញពីរបៀបអានខ្លឹមសាររបស់ file មួយ រួចហើយចម្លងខ្លឹមសារដូចគ្នាទៅឲ្យ file មួយទៀត។

```
#include <stdio.h>
#include <conio.h>
void main(){
   char ch;
   FILE *fp1;
   FILE *fp2;
   /* Assume this test.c file has some data.
      For example "Hi, How are you?" */
   if (fp1 = fopen("test.c", "r")){
      ch = getc(fp1);
      printf("%c", ch);  // display on screen
      // Assume this test2.c file is empty
      fp2 = fopen("test2.c", "w+");
```

```
        while(ch != EOF){
            fputc(ch, fp2);
            ch = getc(fp1);
            printf("%c", ch); // display on screen
        }
        fclose(fp1);
        fclose(fp2);
        getch();
        return;
    }
    getch();
}
```

៧.៧ ការប្រើអនុគមន៍ getc( ) និង putc( )

អនុគមន៍ទាំងពីរនេះត្រូវបានប្រើជាមួយ files ដែលប្រើសម្រាប់អានមួយតួអក្សរពី file មួយ (getc) និង បង្ហាញលទ្ធផល ឬសរសេរតួអក្សរទៅក្នុង file (putc)។ វាមានទម្រង់ទូទៅដូចខាងក្រោម ៖

```
int getc(FILE *fp)
int putc(int char, FILE *fp)
```

<u>ឧទាហរណ៍</u> ៖

```
getc (fp);
putc(char, stdout);
putc(char, fp);
```

កម្មវិធីខាងក្រោមនេះ បង្ហាញពីរបៀបអានខ្លឹមសាររបស់ file មួយ។ ឧបមាថា file មួយមានឈ្មោះ "test.c" ដែលមានទិន្នន័យដូចជា "Hi, How are you?" រួចហើយចម្លងខ្លឹមសារដូចគ្នាទៅឲ្យ file មួយទៀត។

```
#include <stdio.h>
#include <conio.h>
void main(){
    char ch;
```

```
    FILE *fp;
    if (fp = fopen("test.c", "r")){
      ch = getc(fp);
      while (ch != EOF) {
         putc(ch, stdout);
         ch = getc(fp);
      }
      fclose(fp);
      getch();
    }
}
```

៧.៨ ការប្រើអនុគមន៍ feof( )

អនុគមន៍នេះប្រើសម្រាប់រកទីបញ្ចប់នៃ file ។ វាមានទម្រង់ទូទៅដូចខាងក្រោម ៖

int feof(FILE *fp)

<u>ឧទាហរណ៍</u> ៖

feof(fp);

ដែល fp គឺជា file pointer ។

កម្មវិធីខាងក្រោមនេះបង្ហាញពីរបៀបអានខ្លឹមសាររបស់ file មួយ។ ឧបមាថា file មួយមានឈ្មោះ "test.c" ដែលមានទិន្នន័យដូចជា "Hi, How are you?" ។ ការអានទិន្នន័យក្នុង file នេះដោយប្រើអនុគមន៍ fgetc( )។ អនុគមន៍ feof( ) ត្រូវបានប្រើដើម្បីផ្ទៀងផ្ទាត់មើលថាតើវាដល់ទីបញ្ចប់នៃ file ឬនៅ។

```
#include <stdio.h>
#include <conio.h>
void main(){
   FILE *fp ;
   char c ;
   printf("Opening the file test.c in read mode.\n");
```

```
   fp = fopen ("test.c", "r"); // opening an existing file
   if (fp == NULL) {
     printf("Could not open file test.c\n");
     return ;
   }
   printf("Reading the file test.c\n");
   while (1){
     c = fgetc(fp); // reading the file
     if(feof(fp))
        break ;
     printf("%c", c);
   }
   printf("Closing the file test.c as end of file is
reached.");
   fclose(fp); // Closing the file
   getch();
}
```

៧.៩ ការប្រើអនុគមន៍ fgets( )

អនុគមន៍នេះត្រូវបានប្រើសម្រាប់អាន file មួយម្តងមួយឃ្លាៗ។ ទម្រង់ទូទៅរបស់វា គឺ៖

char *fgets(char *string, int n, FILE *fp)

<u>ឧទាហរណ៍</u> ៖

fgets (buffer, size, fp);

ដែលក្នុង buffer គឺជា buffer សម្រាប់ដាក់ទិន្នន័យចូល។ រីង size គឺទំហំរបស់ buffer ។ ចំណែក fp គឺជា file pointer ។

កម្មវិធីខាងក្រោមនេះបង្ហាញពីរបៀបអានខ្លឹមសាររបស់ file មួយ។ ឧបមាថា file មួយមានឈ្មោះ "test.c" ដែលមានទិន្នន័យដូចជា "Hi, How are you?" ។ ការអានទិន្នន័យក្នុង file នេះដោយប្រើអនុគមន៍ fgets( )។

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

```
#include <conio.h>
void main(){
  FILE *fp ;
  char data[50] ;
  printf("Opening the file test.c in read mode\n");
  fp = fopen("test.c", "r") ;
  if ( fp == NULL ){
     printf( "Could not open file test.c\n");
     getch();
     return;
  }
  printf("Reading the file test.c\n");
  while(fgets(data, 50, fp) != NULL)
    printf("%s" , data);
  printf("Closing the file test.c\n") ;
  fclose(fp) ;
  getch();
}
```

៧.១០ ការប្រើអនុគមន៍ fputs( )

អនុគមន៍នេះប្រើសម្រាប់សរសេរ string ទៅដាក់ក្នុង file មួយដែលចង្អុលប្រាប់ដោយ fp។ នៅក្នុងកម្មវិធី C គេសរសេរ string ទៅក្នុង file តាមទម្រង់ខាងក្រោម ៖

int fputs (const char *string, FILE *fp)

<u>ឧទាហរណ៍</u> ៖

fputs (fp, "some data");

កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើអនុគមន៍ fputs( ) ដើម្បីសរសេរ string ចូលទៅក្នុង file មួយឈ្មោះ "test.c" រហូតដល់គេមិនបញ្ចូលអ្វីសោះ ទើបវាបញ្ចប់ការសរសេរ file។

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

```
#include <string.h>
void main(){
    FILE *fp ;
    char data[50];
    // opening an existing file
    printf("Opening the file test.c in write mode\n");
    fp = fopen("test.c", "w") ;
    if (fp == NULL) {
       printf("Could not open file test.c\n");
       getch();
       return;
    }
    printf("\nEnter some text from keyboard" \
           " to write in the file test.c\n");
    // getting input from user
    while (strlen(gets(data)) > 0){
       // writing in the file
       fputs(data, fp) ;
       fputs("\n", fp) ;
    }
    // closing the file
    printf("Closing the file test.c\n") ;
    fclose(fp) ;
    getch();
}
```

៧.១១ ការប្រើអនុគមន៍ fread( ) និង fwrite( )

អនុគមន៍ fread() និង fwrite() ត្រូវបានប្រើសម្រាប់អានខ្លឹមសារពី file ឬ សរសេរទិន្នន័យទៅកាន់ file ដែលបើកដោយប្រើអនុគមន៍ fopen() ។ វាមានទម្រង់ដូចខាងក្រោម ៖

```
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
```

size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);

អនុគមន៍ទាំងពីរនេះទទួលយក arguments បី។ argument ទី 1 គឺជា pointer ចង្អុលប្រាប់ទៅកាន់ buffer ប្រើសម្រាប់ អាន ឬសរសេរទិន្នន័យ។ ទិន្នន័យដែលត្រូវអានឬសរសេរ គឺស្ថិតនៅក្រោមទម្រង់របស់ធាតុ nmemb ដែលធាតុនិមួយៗមានទំហំ size គិតជា bytes។

```
#include <stdio.h>
#include <conio.h>
void main(){
    FILE *fp;
    short x[10] = {1,2,3,4,5,6,5000,6,-10,11};
    short result[10];
    int i;
    fp = fopen("temp.bin", "w+");
    if (fp != NULL){
        fwrite(x, sizeof(short), 10 /*20/2*/, fp);
        rewind(fp);
        fread(result, sizeof(short), 10 /*20/2*/, fp);
    } else
        return;
    printf("Result\n");
    for (i = 0; i < 10; i++)
        printf("%d = %d\n", i, (int)result[i]);
    fclose(fp);
    getch();
}
```

ឧទាហរណ៍កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើអនុគមន៍ fwrite() ដើម្បីសរសេរទិន្នន័យចំនួន 10 លេខ ចូលទៅក្នុង file តាមរយៈ member មួយរបស់ struct Rec ដែលមានឈ្មោះ x។

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

```
struct Rec {
   int x, y, z;
};
void main()    {
   int counter;
   FILE *fp;
   struct Rec myrecord;
   fp = fopen("test.bin", "wb");
   if (!fp) {
      printf("Unable to open file!\n");
      getch();
      return;
   }
   for (counter = 1; counter <= 10; counter++) {
      myrecord.x = counter;
      fwrite(&myrecord, sizeof(struct Rec), 1, fp);
   }
   fclose(fp);
   getch();
}
```

ឧទាហរណ៍កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើអនុគមន៍ fread() ដើម្បីអានទិន្នន័យចេញពី file តាមរយៈ member មួយរបស់ struct Rec ដែលមានឈ្មោះ x។

```
#include <stdio.h>
#include <conio.h>
/* Our structure */
struct Rec {
   int x,y,z;
};
void main() {
   int counter;
```

```
    FILE *fp;
    struct Rec myrecord;
    fp=fopen("test.bin", "rb");
    if (!fp){
       printf("Unable to open file!\n");
       getch();
       return;
    }
    for (counter=1; counter <= 10; counter++) {
       fread(&myrecord, sizeof(struct Rec), 1, fp);
       printf("%d\n", myrecord.x);
    }
    fclose(fp);
    getch();
}
```

៧.១២ ការប្រើអនុគមន៍ fseek( ) **និង** SEEK_SET, SEEK_CUR, SEEK_END

អនុគមន៍នេះត្រូវបានប្រើសម្រាប់ផ្លាស់ទីពីទីតាំង file pointer ទៅកាន់ទីតាំងដែលគេឲ្យ។ ទម្រង់ទូទៅរបស់វា គឺ ៖

int fseek(FILE *fp, long int offset, int whence)

ដែលក្នុងនេះ fp គឺជា file pointer។ offset គឺជាចំនួន bytes / characters ដែលត្រូវផ្លាស់ទីពីទីណាមួយ ឬ ទីតាំង file pointer កំពុងឈរ។ whence គឺជាទីតាំង file pointer កំពុងឈរពីកន្លែង offset ដែលត្រូវបន្ថែម។

- SEEK_SET គឺជាតម្លៃថេរដែលប្រើសម្រាប់ផ្លាស់ទីពីទីតាំង file pointer កំពុងឈរទៅកាន់ខាងដើមនៃ file។
- SEEK_CUR គឺជាតម្លៃថេរដែលប្រើសម្រាប់ផ្លាស់ទីពីទីតាំង file pointer កំពុងឈរទៅកាន់ទីតាំងដែលគេឲ្យ។
- SEEK_END គឺជាតម្លៃថេរដែលប្រើសម្រាប់ផ្លាស់ទីពីទីតាំង file pointer ទៅកាន់ខាងចុងនៃ file។

ឧបមាថា គេមាន file មួយឈ្មោះ test.c ដែលមានទិន្នន័យ "Royal University of Phnom Penh is the best university in Cambodia" ។ ចូរពិនិត្យពីរបៀបផ្លាស់ទី file pointer ទៅកាន់ទីតាំងផ្សេងគ្នា នៅក្នុង file នៃកម្មវិធីខាងក្រោមនេះ ៖

```
#include <stdio.h>
#include <conio.h>
void main(){
   FILE *fp;
   char data[66];
   fp = fopen("mytext.txt", "r");
   fgets(data, 66, fp);
   printf("Before fseek - %s\n", data);

   // To set file pointer to 19th byte/character in the file
   fseek(fp, 19, SEEK_SET);
   fgets (data, 66, fp);
   printf("After SEEK_SET to 19 : %s\n", data);

   // To find backward 11 bytes from current position
   fseek(fp, -11, SEEK_CUR);
   fgets (data, 66, fp);
   printf("After SEEK_CUR to -11 : %s\n", data);

   // To find 9th byte before the end of file
   fseek(fp, 0, SEEK_END);
   fgets (data, 66, fp);
   printf("After SEEK_END to -9 : %s\n", data);

   // To set file pointer to the beginning of the file
   fseek(fp, 0, SEEK_SET); // We can use rewind(fp); also
   fgets(data, 66, fp);
   printf("After setting SEEK_SET : %s\n", data);
   fclose(fp);
   getch();
```

```
}
```

កម្មវិធីខាងក្រោមបង្ហាញឲ្យឃើញពីការប្រើអនុគមន៍ fseek() ដើម្បីផ្លាស់ទី file pointer ទៅតាមចំនួន bytes / characters ដែលបានកំណត់។

```
#include <stdio.h>
#include <conio.h>
/* Our structure */
struct Rec {
  int x,y,z;
};
void main() {
   int counter;
   FILE *fp;
   struct Rec myrecord;
   fp=fopen("test.bin","rb");
   if (!fp) {
      printf("Unable to open file!\n");
      getch();
      return ;
   }
   for ( counter=9; counter >= 0; counter--){
      fseek(fp, sizeof(struct Rec) * counter, SEEK_SET);
      fread(&myrecord, sizeof(struct Rec), 1 , fp);
      printf("%d\n", myrecord.x);
   }
   fclose(fp);
   getch();
}
```

៧.១៣ ការប្រើអនុគមន៍ rewind( )

អនុគមន៍នេះប្រើសម្រាប់កំណត់ទីតាំង file ទៅកាន់ដើម file របស់ stream ដែលគេឲ្យ។ ទម្រង់ទូទៅរបស់វា គឺ ៖

void rewind(FILE *stream)

ដែលក្នុងនេះ stream គឺជា file pointer ។

<u>ឧទាហរណ៍ទី</u> ១ ៖ កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើអនុគមន៍ rewind() ដើម្បីឲ្យ file pointer ទៅកាន់ផ្នែកខាងដើមនៃ file ឈ្មោះ file.txt ។

```
#include <stdio.h>
#include <conio.h>
void main(){
   char str[] = "Royal University of Phnom Penh";
   FILE *fp;
   int ch;
   /* First let's write some content in the file */
   fp = fopen("file.txt", "w");
   fwrite(str, 1, sizeof(str), fp);
   fclose(fp);
   fp = fopen("file.txt", "r");
   while(1){
      ch = fgetc(fp);
      if (feof(fp)) {
         break ;
      }
      printf("%c", ch);
   }
   rewind(fp);
   printf("\n");
   while(1){
      ch = fgetc(fp);
```

```
      if(feof(fp)){
         break ;
      }
      printf("%c", ch);
   }
   fclose(fp);
   getch();
}
```

<u>ឧទាហរណ៍ទី ២</u> ៖ កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើអនុគមន៍ rewind() ដើម្បីឲ្យ file pointer ទៅកាន់ផ្នែកខាងដើមនៃ binary file ឈ្មោះ test.bin ។

```
#include <stdio.h>
#include <conio.h>
/* Our structure */
struct Rec {
   int x, y, z;
};
void main(){
   int counter;
   FILE *fp;
   struct Rec myrecord;
   fp = fopen("test.bin", "rb");
   if (!fp) {
      printf("Unable to open file!\n");
      getch();
      return;
   }
   fseek(fp, sizeof(struct Rec), SEEK_END);
   rewind(fp);
   for ( counter=1; counter <= 10; counter++){
      fread(&myrecord, sizeof(struct Rec), 1, fp);
```

```
        printf("%d\n",myrecord.x);
    }
    fclose(fp);
    getch();
}
```

៧.១៤ ការប្រើអនុគមន៍ ftell( )

អនុគមន៍នេះត្រូវបានប្រើសម្រាប់រកទីតាំង file pointer នៅក្នុង file ដោយអនុលោមតាមការចាប់ផ្តើមនៃ file ។

ទម្រង់ទូទៅរបស់វាគឺ ៖

long ftell(FILE *stream)

ដែលក្នុងនេះ stream គឺជា pointer ចង្អុលទៅកាន់ FILE object ដែលដូចគ្នាទៅនឹង stream។

អនុគមន៍នេះឲ្យតម្លៃនៃការបញ្ជាក់ប្រាប់ទីតាំងកំពុងឈរ។ បើសិនជាមាន error មួយកើតឡើងនោះវានឹងឲ្យតម្លៃ -1L និងអញ្ជាតប្រភេទ global errno ត្រូវកំណត់តម្លៃវិជ្ជមាន។

ឧទាហរណ៍កម្មវិធីខាងក្រោមនេះបង្ហាញពីការប្រើអនុគមន៍ ftell() ដើម្បីដឹងពីចំនួន bytes របស់ file "test.txt"។

```
#include <stdio.h>
#include <conio.h>
void main(){
    FILE *fp;
    int len;
    fp = fopen("test.txt", "r");
    if( fp == NULL ) {
        perror ("Error opening file");
        getch();
        return;
    }
    fseek(fp, 0, SEEK_END);
    len = ftell(fp);
    fclose(fp);
```

```
    printf("Total size of file.txt = %d bytes\n", len);
    getch();
}
```

៧.១៥ ការប្រើអនុគមន៍ fscanf( )

អនុគមន៍នេះត្រូវបានប្រើសម្រាប់អានទិន្នន័យពីទីតាំងកំពុងប្រើនៃ stream ដែលបានកំណត់ទៅកាន់ទីតាំងដែលគេឲ្យតាមរយៈការបញ្ជូល argument-list។ ការបញ្ជូលនិមួយៗក្នុង argument-list ត្រូវតែជា pointer មួយដែលចង្អុលទៅកាន់អញ្ញាតមួយដោយមានប្រភេទទិន្នន័យមួយដែលឆ្លើយតបទៅនឹងការបញ្ជាក់ប្រភេទទិន្នន័យនៅក្នុង format-string។

ទម្រង់ទូទៅរបស់វា គឺ៖

int fscanf (FILE *stream, const char *format-string, argument-list);

<u>ឧទាហរណ៍ទី ១</u> ៖ កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើអនុគមន៍ fscanf() ដែលអាចអានតម្លៃចេញពី file មួយបាន។

```
#include <stdio.h>
#include <conio.h>
void main(){
   FILE *fp;
   int roll;
   char name[25];
   float marks;
   fp = fopen("file2.txt","r");  //Statement 1
   if(fp == NULL){
      printf("\nCan't open file or file doesn't exist.");
      getch();
      return;
   }
   printf("\nData in file...\n");
   fscanf(fp,"%d %s %f", &roll, name, &marks);
   printf("\n%d\t%s\t%f", roll, name, marks);
```

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

<u>ឧទាហរណ៍ទី ២</u> ៖ កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើអនុគមន៍ fscanf() ដែលអាចអាន text រហូតដល់ជួបតួអក្សរចុះបន្ទាត់ (\n) ចេញពី file មួយ។

```
#include <stdio.h>
#include <conio.h>
void main() {
    char c[1000];
    FILE *fp;
    if ((fp = fopen("test.txt", "r")) == NULL) {
        printf("Error! opening file");
        // Program exits if file pointer returns NULL.
        getch();
        return;
    }
    // reads text until newline
    fscanf(fp,"%[^\n]", c);
    printf("Data from the file:\n%s", c);
    fclose(fptr);
    getch();
}
```

<u>ឧទាហរណ៍ទី ៣</u> ៖ កម្មវិធីខាងក្រោមបង្ហាញពីការប្រើអនុគមន៍ fscanf() ដែលអាចអាន text រហូតដល់ទីបញ្ចប់នៃ file (EOF) ចេញពី file មួយ។

```
#include <stdio.h>
#include <conio.h>
void main() {
```

```
    char c[1000];
    FILE *fp;
    if ((fp = fopen("test.txt", "r")) == NULL) {
        printf("Error! opening file");
        // Program exits if file pointer returns NULL.
        getch();
        return;
    }
    // read text until EOF
    fscanf(fp, "%[^\EOF]", c);
    printf("Data from the file:\n%s", c);
    fclose(fptr);
    getch();
}
```