CSCI2100C Tutorial

ZHANG Xinyun

xyzhang21@cse.cuhk.edu.hk

- Symbol table exercise
- File I/O

Exercise1

Print the keys

Problem definition

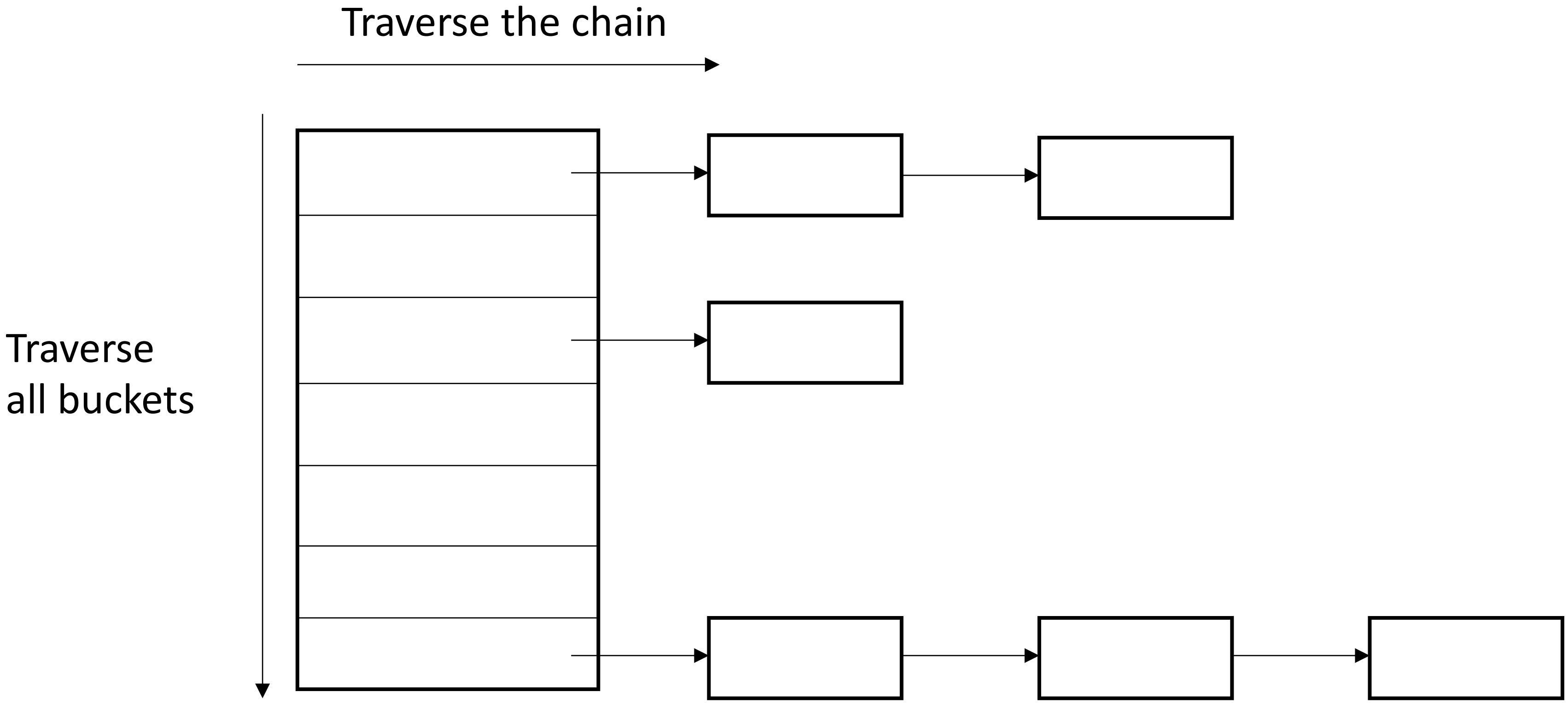
Write the C function PrintKeys() in implementation file to print all the keys in a symbol table. The function accepts a symtabADT argument, use the following function prototype. Note that this function should be implemented in symtab.c and the prototype should be included in symtab.h.

```
#include “symtab.h”
void PrintKeys(symtabADT table);
```

Key	Value
“a”	3
“b”	2
“c”	2
“d”	2

```
Out: “a” “b” “c” “d”
```

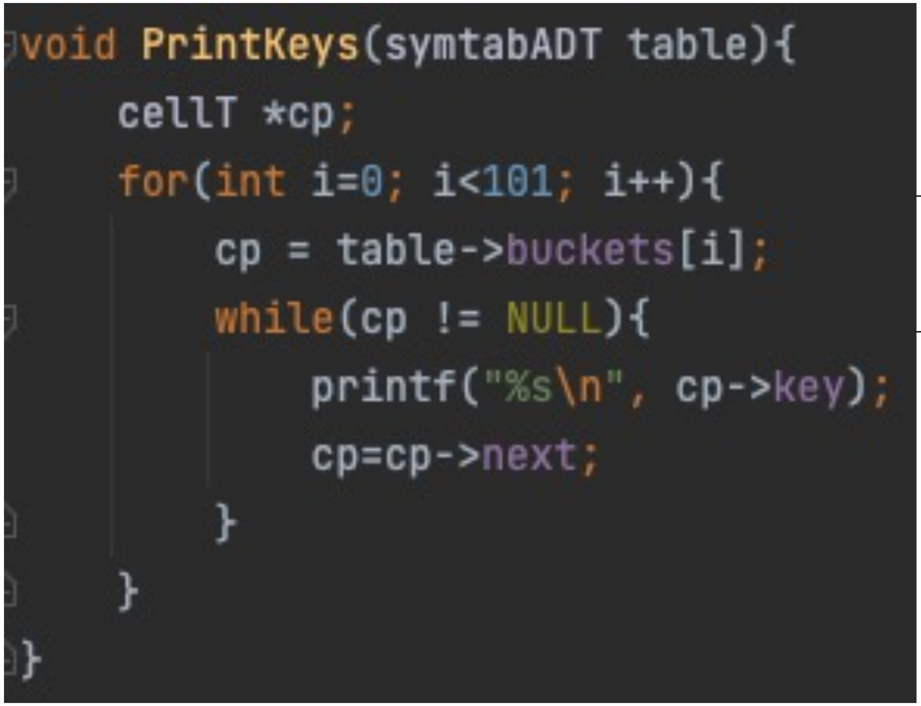
Answer



Traverse the chain

Traverse all buckets

Answer



```
void PrintKeys(symtabADT table){
    cellT *cp;
    for(int i=0; i<101; i++){
        cp = table->buckets[i];
        while(cp != NULL){
            printf("%s\n", cp->key);
            cp=cp->next;
        }
    }
}
```

// Traverse the bucket

// Traverse the chain

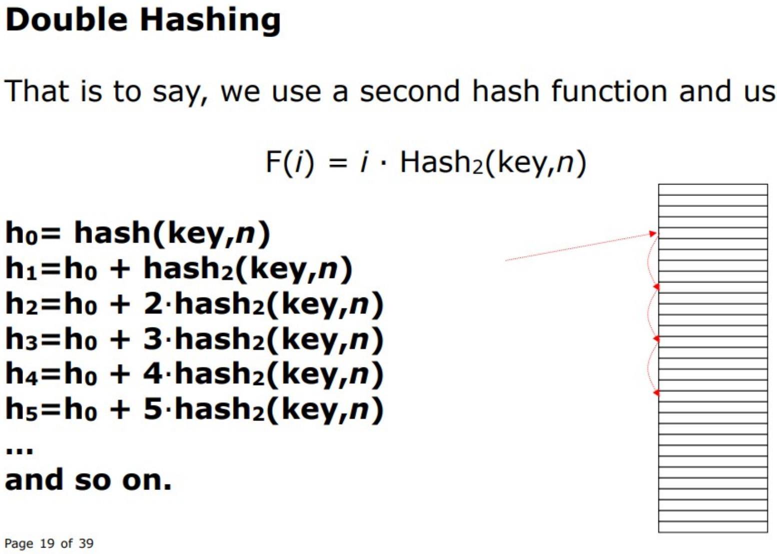
Exercise2

Hash table

Problem Definition

Assume Table size = 10 elements
Hash1(key) = key % 10
Hash2(key) = 7 – (key % 7)
Insert keys: 89, 18, 49, 58, 69

Write the hash of each key using Open Addressing Hashing (double hashing).



Double Hashing

That is to say, we use a second hash function and use

$$F(i) = i \cdot \mathrm{Hash}_2(\mathrm{key}, n)$$

h_0= hash(key,n)
h_1=h_0 + hash$_2$(key,n)
h_2=h_0 + 2·hash$_2$(key,n)
h_3=h_0 + 3·hash$_2$(key,n)
h_4=h_0 + 4·hash$_2$(key,n)
h_5=h_0 + 5·hash$_2$(key,n)
...
and so on.

Page 19 of 39

Answer

- Hash(89) = Hash1(89) = 89 % 10 = **9**
- Hash(18) = Hash1(18) = 18 % 10 = **8**
- Hash(49) = Hash1(49) = 49 % 10 = 9, a collision!

  = Hash1(49) + Hash2(49) = 9 + 7 – (49 % 7) = 16 -> 16 % 10 = **6**
- Hash(58) = Hash1(58) = 58 % 10 = 8, a collison!

  = Hash1(58) + Hash2(58) = 8 + 7 – (58 % 7) = 13 -> 13 % 10 = **3**
- Hash(69) = Hash1(69) = 69 % 10 = 9, a collision!

  = Hash1(69) + Hash2(69) = 9 + 7 – (69 % 7) = 10 -> 10 % 10 = **0**

Exercise3

Two Sum

Problem Definition

Given an array of integers *nums*, and an integer *target*, return indices of the two numbers such that they add up to *target*. You may assume that each input would have exactly one solution, and you may not use the same element twice.

E.g.
Input: nums = [2, 7, 11, 15], target = 9
Output: [0, 1]
Output: Because nums[0] + nums[1] == 9, we return [0, 1].

Answer – Brute force

nums = [2, 7, 11, 15]   target = 9

Residual = 9 – 2 = 7 -> Look for 7 in the array nums (Linear search)
Residual = 9 – 7 = 2 -> Look for 2 in the array nums (Linear search)
Residual = 9 – 11 = -2 -> Look for -2 in the array nums (Linear search)
Residual = 9 – 15 = -6 -> Look for -6 in the array nums (Linear search)

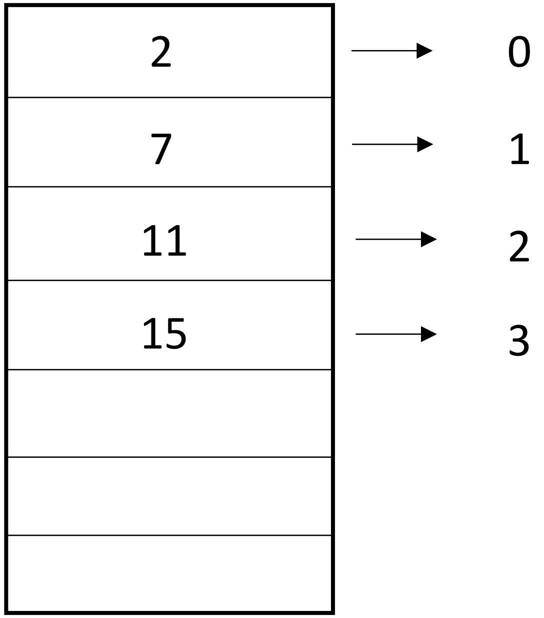
Answer – Hash table

nums = [2, 7, 11, 15]
Index: 0 1 2 3

target = 9

Residual = 9 – 2 = 7 -> Look for 7 in the Hash table -> index0
Residual = 9 – 7 = 2 -> Look for 2 in the Hash table -> index1
Residual = 9 – 11 = -2 -> Look for -2 in the Hash table -> NULL
Residual = 9 – 15 = -6 -> Look for -6 in the Hash table -> NULL

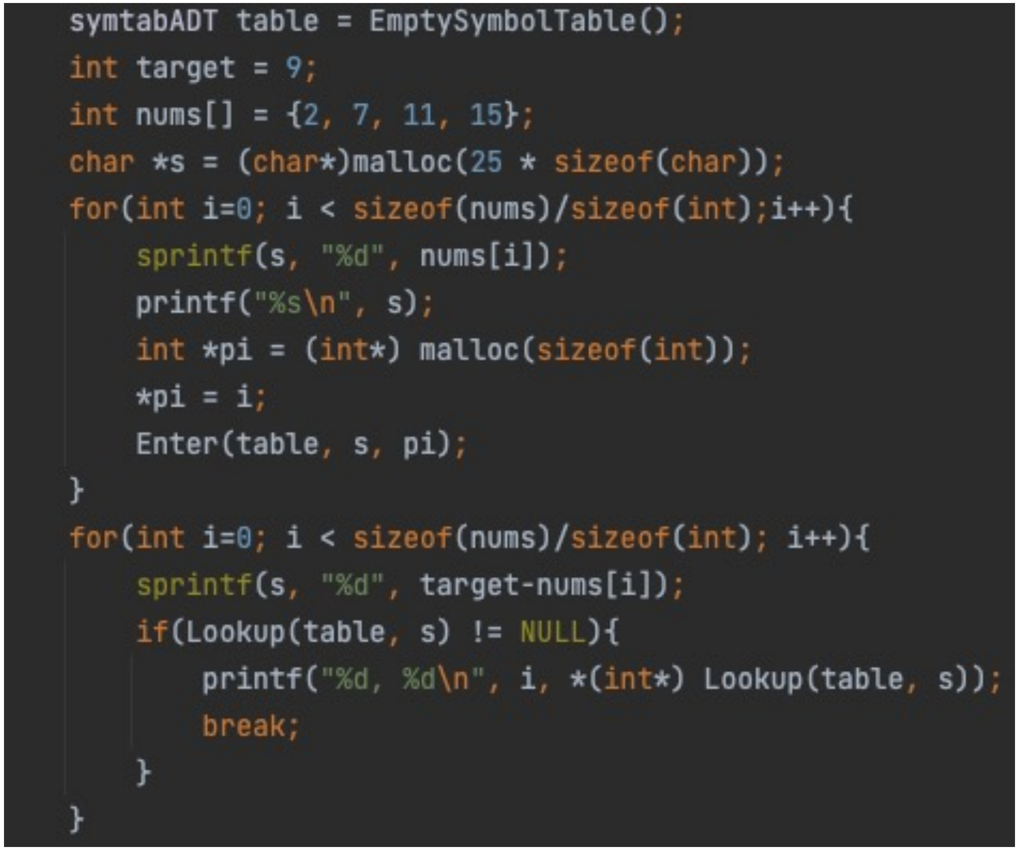
Keys: Elements in the array
Value: Index



2 → 0
7 → 1
11 → 2
15 → 3

Brute force consumes n^2 times search.
Hash table consumes n times search.

Answer – Hash table



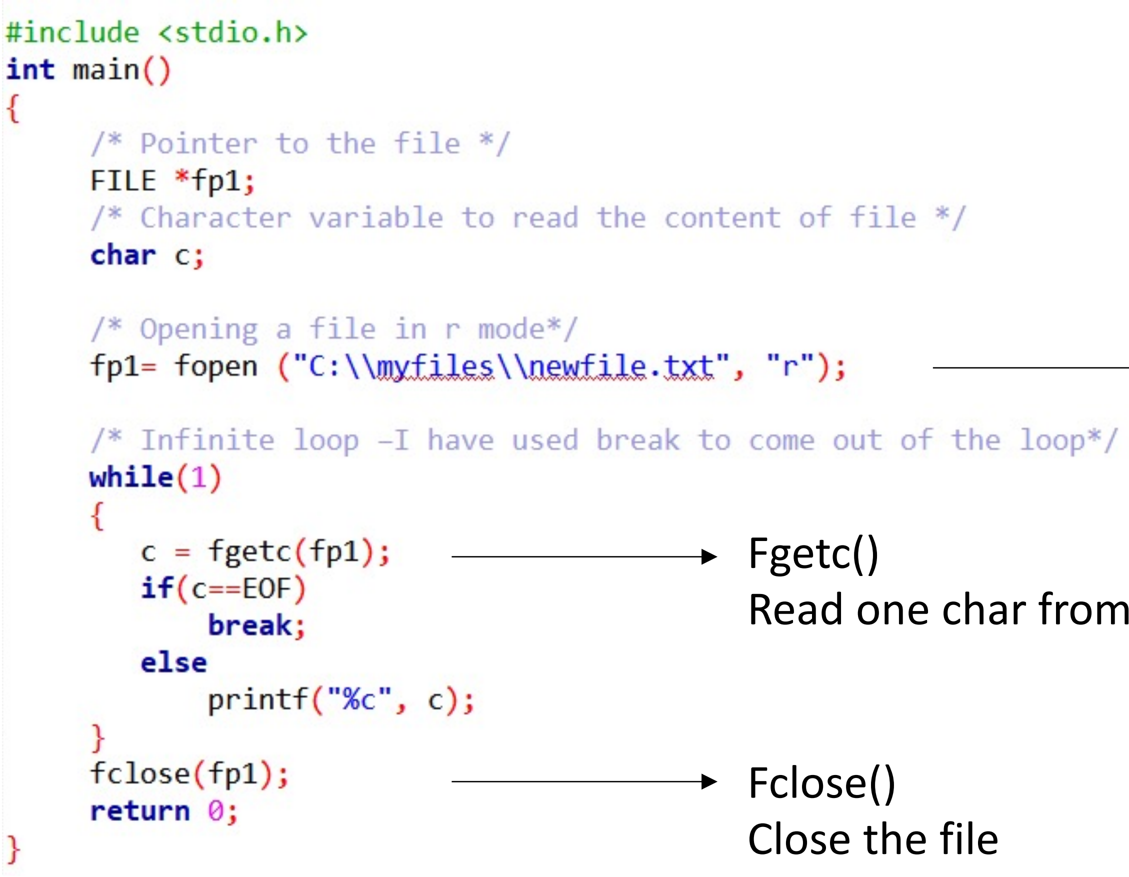
```
symtabADT table = EmptySymbolTable();
int target = 9;
int nums[] = {2, 7, 11, 15};
char *s = (char*)malloc(25 * sizeof(char));
for(int i=0; i < sizeof(nums)/sizeof(int);i++){
    sprintf(s, "%d", nums[i]);
    printf("%s\n", s);
    int *pi = (int*) malloc(sizeof(int));
    *pi = i;
    Enter(table, s, pi);
}
for(int i=0; i < sizeof(nums)/sizeof(int); i++){
    sprintf(s, "%d", target-nums[i]);
    if(Lookup(table, s) != NULL){
        printf("%d, %d\n", i, *(int*) Lookup(table, s));
        break;
    }
}
```

//Enter Table

//Search Table
(Linear time)

FILE I/O

File I/O



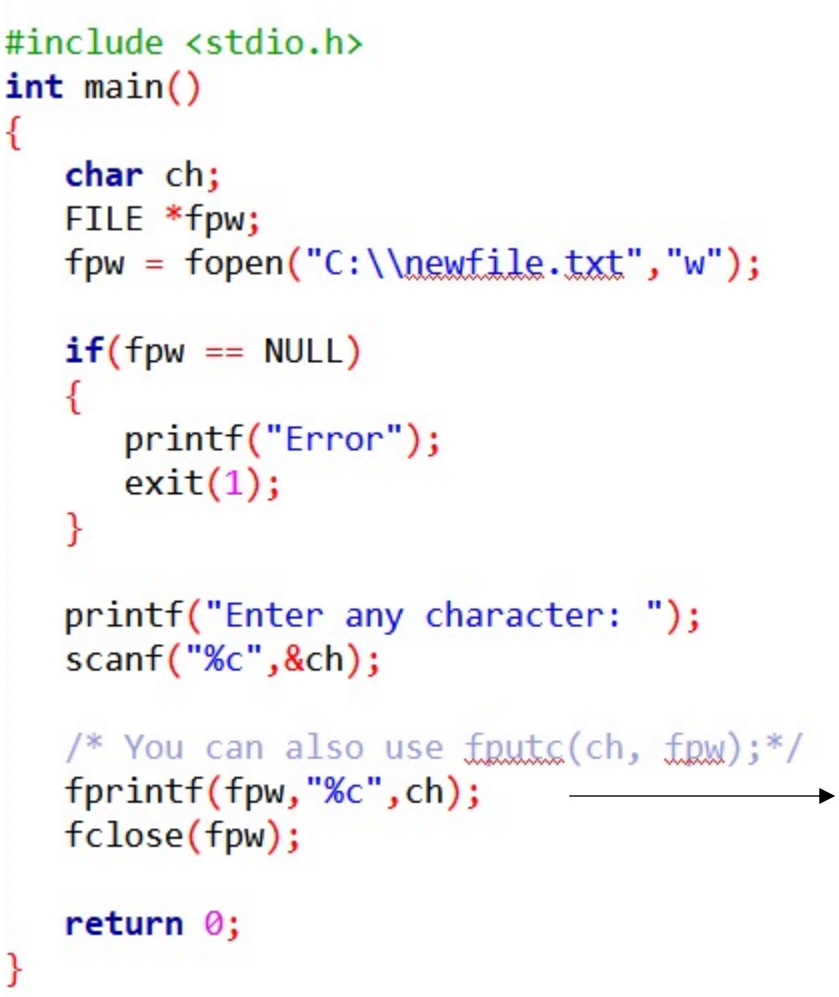
```
#include <stdio.h>
int main()
{
     /* Pointer to the file */
     FILE *fp1;
     /* Character variable to read the content of file */
     char c;

     /* Opening a file in r mode*/
     fp1= fopen ("C:\\myfiles\\newfile.txt", "r");

     /* Infinite loop -I have used break to come out of the loop*/
     while(1)
     {
        c = fgetc(fp1);
        if(c==EOF)
            break;
        else
            printf("%c", c);
     }
     fclose(fp1);
     return 0;
}
```

Fopen(“file_name”, “Mode”);
“r”: read only
“w”: write only
“a”: read + write

Fgetc()
Read one char from current position

Fclose()
Close the file

File I/O

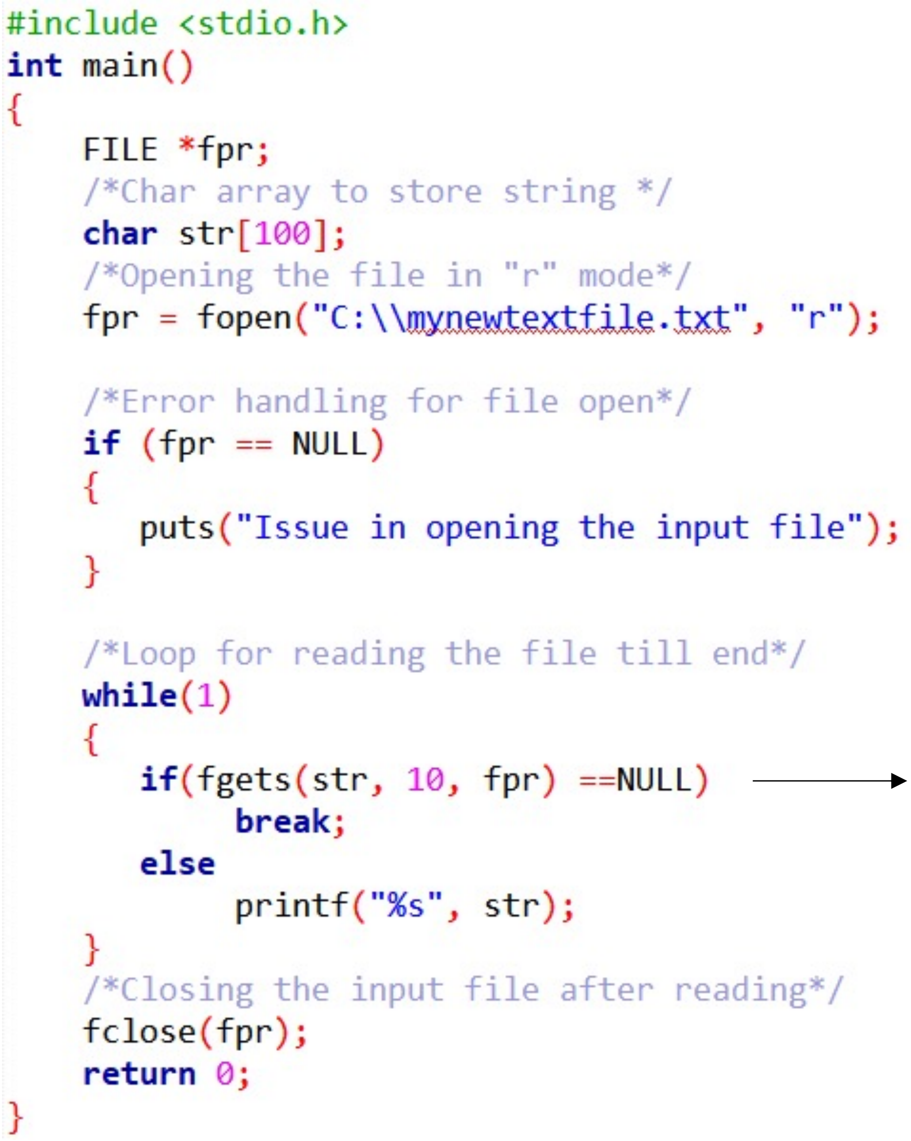


```
#include <stdio.h>
int main()
{
    char ch;
    FILE *fpw;
    fpw = fopen("C:\\newfile.txt","w");

    if(fpw == NULL)
    {
        printf("Error");
        exit(1);
    }

    printf("Enter any character: ");
    scanf("%c",&ch);

    /* You can also use fputc(ch, fpw);*/
    fprintf(fpw,"%c",ch);
    fclose(fpw);

    return 0;
}
```

fprintf() writes chars into the file.

fprintf(fpw, “%s”, ch);
// write a string if ch is an array

File I/O

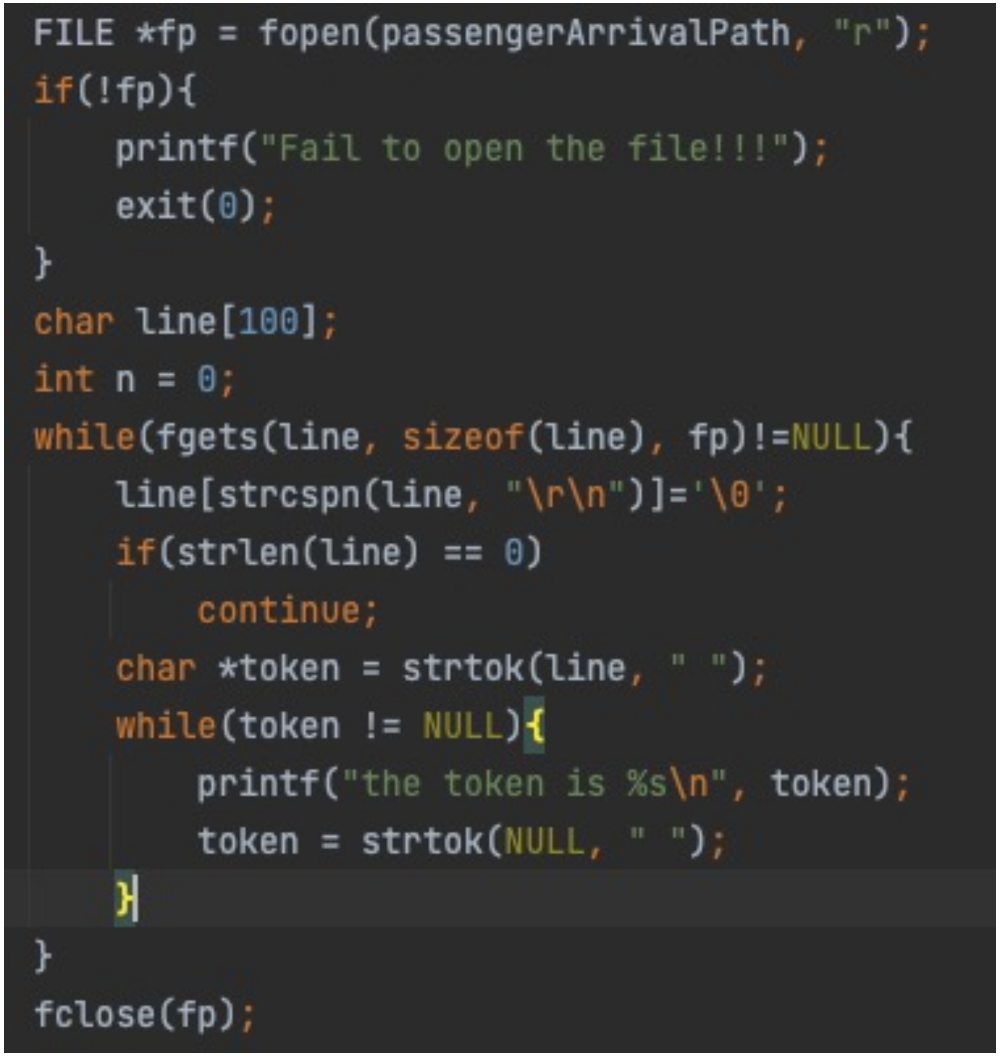


```
#include <stdio.h>
int main()
{
    FILE *fpr;
    /*Char array to store string */
    char str[100];
    /*Opening the file in "r" mode*/
    fpr = fopen("C:\\mynewtextfile.txt", "r");

    /*Error handling for file open*/
    if (fpr == NULL)
    {
       puts("Issue in opening the input file");
    }

    /*Loop for reading the file till end*/
    while(1)
    {
       if(fgets(str, 10, fpr) ==NULL)
            break;
       else
            printf("%s", str);
    }
    /*Closing the input file after reading*/
    fclose(fpr);
    return 0;
}
```

→

char *fgets(char *s, int rec_len, FILE *fpr);
s: string
rec_len: input length
fpr: file ptr

File I/O



```c
FILE *fp = fopen(passengerArrivalPath, "r");
if(!fp){
    printf("Fail to open the file!!!");
    exit(0);
}
char line[100];
int n = 0;
while(fgets(line, sizeof(line), fp)!=NULL){
    line[strcspn(line, "\r\n")]='\0';
    if(strlen(line) == 0)
        continue;
    char *token = strtok(line, " ");
    while(token != NULL){
        printf("the token is %s\n", token);
        token = strtok(NULL, " ");
    }
}
fclose(fp);
```

```
400 1 12
500 12 12
400 9 8
77 32 12
```