C Programming Lecture 16

Wednesday, 26 June 2024 8:18 PM

Strings in C

The string can be defined as the one-dimensional array of characters terminated by a null ('\0'). The character array or the string is used to manipulate text such as word or sentences.

Each character in the array occupies one byte of memory, and the last character must always be 0. The termination character ('\0') is important in a string since it is the only way to identify where the string ends.

When we define a string as char s[10], the character s[10] is implicitly initialized with the null in the memory.

```c
#include<stdio.h>
void main(){
    char s[] = "Hello";
    printf("%d", sizeof(s));
}
```

6B Hello\0

```c
#include<stdio.h>
void main(){
    char c1[] = "Hello";
    char c2[] = {'H', 'e', 'l', 'l', 'o'};
    char* c3 = "Hello";
    printf("%s \n", c1);
    printf("%s \n", c2);
    printf("%s \n", c3);
}
```

```c
#include<stdio.h>
void main(){
    char* c1 = "Hello";
    char* p = c1;
    char* c2;
    while(*p) {
        *c2++ = *p++;
    }
    printf("%s", c2);
}
```

gets() and puts()

scanf printf

- The gets() function enables the user to enter some characters followed by the enter key.
- All the characters entered by the user get stored in a character array.
- The null character is added to the array to make it a string.
- The gets() allows the user to enter the space-separated strings.
- It returns the string entered by the user.

```
char a[15];
gets(a);
```

```
char[] gets(char[]);
```

- The puts() function is very much similar to printf() function.
- The puts() function is used to print the string on the console which is previously read by using gets() or scanf() function.
- The puts() function returns an integer value representing the number of characters being printed on the console

```
int puts(char[])
```

String operations

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

There are many important string functions defined in "string.h" library.

No.	Function	Description
1)	`strlen(string_name)`	returns the length of string name.
2)	`strcpy(destination, source)`	copies the contents of source string to destination string.
3)	`strcat(first_string, second_string)`	concats or joins first string with second string. The result of the string is stored in first string.
4)	`strcmp(first_string, second_string)`	compares the first string with second string. If both strings are same, it returns 0.

char a[10] = "Hello"; strcpy(b, a);

strlen(a); strcat(a, b)

char a[] = "Hello"; ✓
char b[] = "World"; ✓
strcat(a,b); → a = HelloWorld
b = World.

Strcmp(a,b)
↓
Dictionary.

+ve if a > b ✓
-ve if a < b
0 if a == b

Hello
World

Hello(0)
Hell(i)

Hello > Helli

Hello - World
World - Hello
Hello(0) - Hel(i)

```c
#include<stdio.h>
#include<string.h>
int main(){
    char a[6] = "Hello";
    char b[6] = "World";
    char c[12];
    // copies the contents of source string to destination string.
    strcpy(c, a);
    // concats or joins first string with second string.
    strcat(c, b);

    printf("String c: %s \n", c);
    // returns the length of string.
    printf("Length of c: %d \n", strlen(c));
    // returns comparision of strings
    printf("Str A - Str B = %d \n", strcmp(a, b));

    return 0;
}
```

Math Functions in C

There are various methods in math.h header file. The commonly used functions of math.h header file are given below.

No.	Function	Description
1)	ceil(number)	rounds up the given number. It returns the integer value which is greater than or equal to given number.
2)	floor(number)	rounds down the given number. It returns the integer value which is less than or equal to given number.
3)	sqrt(number)	returns the square root of given number.
4)	pow(base, exponent)	returns the power of given number.
5)	abs(number)	returns the absolute value of given number.

```c
#include<stdio.h>
#include<math.h>
int main(){
    printf("\n%f", ceil(3.6));
    printf("\n%f", floor(3.2));
    printf("\n%f", sqrt(7));
    printf("\n%f", pow(2,4));
    printf("\n%d", abs(-12));
    return 0;
}
```

ceil → 4
floor → 3
sqrt →
pow → 16
abs → 12

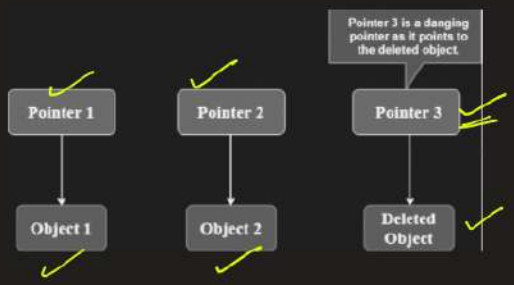
ceil → i/f least integer gr or equal to x

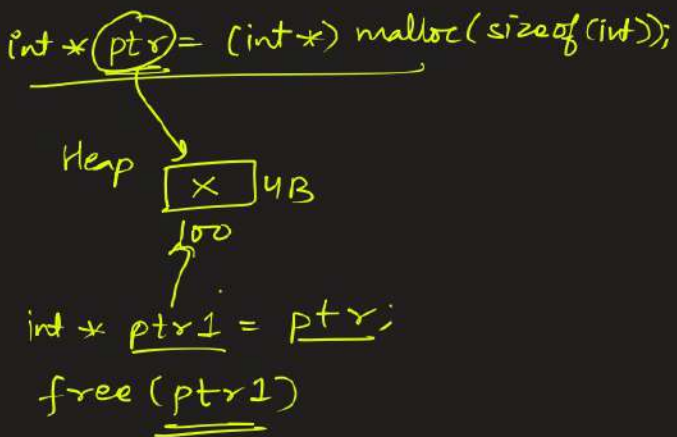
floor → g/f gr. integer less or equal to x

pow(a,b) → a^b

gcd a^b sqrt(n)

Dangling pointers in C





File Handling in C

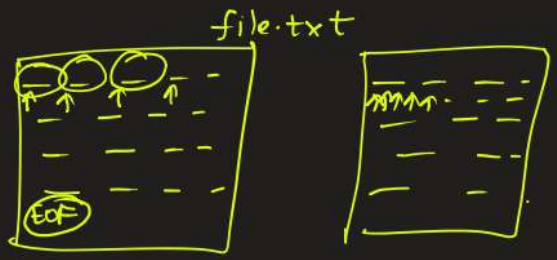
File handling in C enables us to create, update, read, and delete the files stored on the local file system through our C program

There are many functions in the C library to open, read, write, search and close the file. A list of file functions are given below:

No.	Function	Description
1	`FILE *fopen( const char * filename, const char * mode );`	opens new or existing file
2	`int fprintf(FILE *stream, const char *format)`	write data into the file
3	`int fscanf(FILE *stream, const char *format)`	reads data from the file
4	`int fputc(int c, FILE *stream)`	writes a character into the file
5	`char fgetc(FILE *stream)`	reads a character from file
6	`int fclose( FILE *fp );`	closes the file
7	`fseek()`	sets the file pointer to given position
8	`fputw()`	writes an integer to file
9	`fgetw()`	reads an integer from file
10	`ftell()`	returns current position
11	`rewind()`	sets the file pointer to the beginning of the file

fgets()
fputs()

FILE * fp = fopen("file.txt", "r")



r+
w+
} r & w

Strings (10)

[Input] file.txt

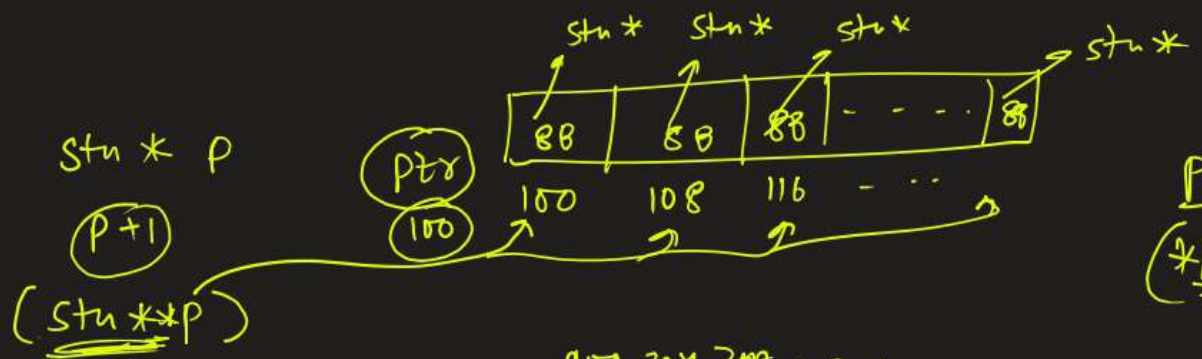
Name	Roll_no	Physics (100)	Chem (100)	Maths (100)
Kapil	1001	34	56	37
Shreya	1002	57	68	82
Rahul	1012	91	67	93
Ishaan	1036	66	52	71
Dhruv	1041	7	11	21

percentage Grade

\t \n

whitespace

fscanf



P[0] → name

(*P[0]).name

P[0] = ptr

P[1] = ptr

(int *p)

(a)

[Program] main.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct{
    char name[20];
    unsigned long long roll;
    int phy;
    int chem;
    int math;
    float per;
    char grade;
} stu;

void print_stu(stu *s){
    printf("Name: %s, Roll: %llu, Phy: %d, Chem: %d, Math: %d , Per: %.2f, Grade: %c \n",
            s->name, s->roll, s->phy, s->chem, s->math, s->per, s->grade);
}
```

```c
int read_file(FILE* fp, stu** ptr, int num_col){
    char val[50];
    int ns = 0, rc = 0;
    for(int i=0; i<num_col; i++)
        fscanf(fp, "%s", val);
    while(fscanf(fp, "%s", val) != EOF) {
        switch(rc) {
            case 0:
                ptr[ns] = (stu*)malloc(sizeof(stu));
                ns++;
                strcpy(ptr[ns-1]->name, val);
                break;
            case 1:
                ptr[ns-1]->roll = (unsignedlonglong)atoi(val);
                break;
            case 2:
                ptr[ns-1]->phy = atoi(val);
                break;
            case 3:
                ptr[ns-1]->chem = atoi(val);
                break;
            case 4:
                ptr[ns-1]->math = atoi(val);
                break;
        }
        if(++rc == num_col)
            rc = 0;
    }
    return ns;
}
```

```c
void analyze(stu** ptr, int ns){
    for(int i=0; i<ns; i++) {
        int p = ptr[i]->phy, c=ptr[i]->chem, m=ptr[i]->math;
        ptr[i]->per = (p+c+m)/3.0;
        if(ptr[i]->per > 80.0)
            ptr[i]->grade = 'A';
        else if(ptr[i]->per>60.0)
            ptr[i]->grade = 'B';
        else if(ptr[i]->per>40.0)
            ptr[i]->grade = 'C';
     else
            ptr[i]->grade = 'F';
    }
}

void write_analysis(stu** ptr, int ns){
    FILE *fp; // file pointer
    fp = fopen("results.csv","w");
    fprintf(fp, "Name,Percentage,Grade\n");
    for(int i=0; i<ns; i++)
        fprintf(fp, "%s,%.2f,%c\n",
            ptr[i]->name, ptr[i]->per, ptr[i]->grade);
    fclose(fp);
}
```

```c
int main(){
    FILE *fp; // file pointer
    fp = fopen("file.txt","r");
    int num_col = 5, rc = 0;
    stu* ptr[10];
    int ns = read_file(fp, ptr, num_col);
    analyze(ptr, ns);
    for(int i=0; i<ns; i++) {
        print_stu(ptr[i]);
    }
    fclose (fp);
    write_analysis(ptr, ns);
    return 0;
}
```

[Output] results.csv

```
Name,Percentage,Grade
Kapil,42.33,C
Shreya,69.00,B
Rahul,83.67,A
Ishaan,63.00,B
Dhruv,13.00,F
```