**ASSIGNMENT 7: STRUCTURE AND FILE**

1. Create a structure to specify data on students given below: Roll number, Name, Department, Course, Year of joining, Marks of 6 Subjects Write a function to print the names of all students who joined in a particular year.
2. Write a function to print the data of a student whose roll number is given.
3. Write a function to create and store this information in a text file.
4. Write a function which prints in ascending order the rank list (Roll, Name, Department, Average) based on the average of 6 subjects.

Code:

#include <stdio.h>

#include <stdlib.h>

#define MAX\_STUDENTS 100

struct student {

int roll\_number;

char name[50];

char department[50];

char course[50];

int year\_of\_joining;

int marks[6];

float average;

};

struct student students[MAX\_STUDENTS];

int num\_students = 0;

// Function to print the names of all students who joined in a particular year.

void print\_students\_by\_year(int year) {

int i;

for (i = 0; i < num\_students; i++) {

if (students[i].year\_of\_joining == year) {

printf("%s\n", students[i].name);

}

}

}

// Function to print the data of a student whose roll number is given.

void print\_student\_by\_roll(int roll) {

int i;

for (i = 0; i < num\_students; i++) {

if (students[i].roll\_number == roll) {

printf("Roll Number: %d\n", students[i].roll\_number);

printf("Name: %s\n", students[i].name);

printf("Department: %s\n", students[i].department);

printf("Course: %s\n", students[i].course);

printf("Year of Joining: %d\n", students[i].year\_of\_joining);

printf("Marks:\n");

int j;

for (j = 0; j < 6; j++) {

printf("%d ", students[i].marks[j]);

}

printf("\n");

return;

}

}

printf("Student not found.\n");

}

// Function to create and store student information in a text file.

void store\_students\_to\_file(char \*filename) {

FILE \*fp;

fp = fopen(filename, "w");

if (fp == NULL) {

printf("Error opening file %s.\n", filename);

return;

}

int i, j;

for (i = 0; i < num\_students; i++) {

fprintf(fp, "%d %s %s %s %d ", students[i].roll\_number, students[i].name, students[i].department, students[i].course, students[i].year\_of\_joining);

for (j = 0; j < 6; j++) {

fprintf(fp, "%d ", students[i].marks[j]);

}

fprintf(fp, "\n");

}

fclose(fp);

}

// Function to input student data.

void input\_students() {

int i;

printf("Enter the number of students to input: ");

scanf("%d", &num\_students);

for (i = 0; i < num\_students; i++) {

printf("Enter data for student %d:\n", i+1);

printf("Roll Number: ");

scanf("%d", &students[i].roll\_number);

printf("Name: ");

scanf("%s", students[i].name);

printf("Department: ");

scanf("%s", students[i].department);

printf("Course: ");

scanf("%s", students[i].course);

printf("Year of Joining: ");

scanf("%d", &students[i].year\_of\_joining);

printf("Marks (6 subjects): ");

int j;

for (j = 0; j < 6; j++) {

scanf("%d", &students[i].marks[j]);

}

}

}

// Function to print the rank list in ascending order based on the average marks of six subjects

void print\_rank\_list()

{

// Calculate the average marks for each student

for (int i = 0; i < num\_students; i++) {

float total = 0;

for (int j = 0; j < 6; j++) {

total += students[i].marks[j];

}

students[i].average = total / 6.0;

}

// Sort the students in ascending order based on the average marks

for (int i = 0; i < num\_students - 1; i++) {

for (int j = i + 1; j < num\_students; j++) {

if (students[i].average > students[j].average) {

// Swap the students

struct student temp = students[i];

students[i] = students[j];

students[j] = temp;

}

}

}

// Print the rank list

printf("Rank\tRoll Number\tName\t\tDepartment\tAverage\n");

for (int i = num\_students-1; i >=0; i--) {

printf("%d\t%d\t\t%s\t\t%s\t\t%.2f\n", num\_students-i, students[i].roll\_number, students[i].name, students[i].department, students[i].average);

}

}

int main() {

input\_students();

int yr,r;

printf("Enter the desired year: ");

scanf("%d",&yr);

print\_students\_by\_year(yr);

store\_students\_to\_file("students.txt");

printf("Enter the desired roll number");

scanf("%d",&r);

print\_student\_by\_roll(r);

print\_rank\_list();

return 0;

}

Output:

Enter the number of students to input: 3

Enter data for student 1:

Roll Number: 1

Name: ab

Department: cse

Course: ug1

Year of Joining: 2022

Marks (6 subjects): 1 2 3 4 5 6

Enter data for student 2:

Roll Number: 2

Name: cd

Department: etce

Course: ug2

Year of Joining: 2021

Marks (6 subjects): 0 0 1 2 3 4

Enter data for student 3:

Roll Number: 3

Name: ef

Department: ee

Course: ug1

Year of Joining: 2022

Marks (6 subjects): 1 1 1 1 1 1

Enter the desired year: 2022

ab

ef

Enter the desired roll number3

Roll Number: 3

Name: ef

Department: ee

Course: ug1

Year of Joining: 2022

Marks:

1 1 1 1 1 1

Rank Roll Number Name Department Average

1 1 ab cse 3.50

2 2 cd etce 1.67

3 3 ef ee 1.00

Students.txt:

1 ab cse ug1 2022 1 2 3 4 5 6

2 cd etce ug2 2021 0 0 1 2 3 4

3 ef ee ug1 2022 1 1 1 1 1 1

1. Write a program in C for following tasks
2. Output of the code is the program itself.
3. Output of the code is the program itself.
4. Reads a C source file and determines the percentage of characters which are part of comments.
5. Reads a C source file and determines no of variables defined of each built in data types.
6. Copy one file to another where files names are passed as command line arguments.

Code:

a)

#include <stdio.h>

int main() {

FILE \*fp;

char c;

fp = fopen(\_\_FILE\_\_, "r");

do {

c = getc(fp);

putchar(c);

} while (c != EOF);

fclose(fp);

return 0;

}

b)

#include <stdio.h>

int main() {

FILE \*fp1, \*fp2;

char c;

fp1 = fopen("input.c", "r");

if (fp1 == NULL) {

printf("Error: input file cannot be opened\n");

return 1;

}

fp2 = fopen("output.c", "w");

if (fp2 == NULL) {

printf("Error: output file cannot be opened\n");

return 1;

}

while ((c = getc(fp1)) != EOF) {

if (c == '/') {

char next\_char = getc(fp1);

if (next\_char == '\*') {

// comment block begins

while (1) {

c = getc(fp1);

if (c == '\*' && getc(fp1) == '/') {

// comment block ends

break;

}

}

} else {

// not a comment, write both characters

putc(c, fp2);

putc(next\_char, fp2);

}

} else {

// not a comment, write character

putc(c, fp2);

}

}

fclose(fp1);

fclose(fp2);

printf("Comments removed and copied to output file successfully\n");

return 0;

}

c)

#include <stdio.h>

int main() {

FILE \*fp;

char c;

int total\_chars = 0, comment\_chars = 0;

fp = fopen("input.c", "r");

if (fp == NULL) {

printf("Error: input file cannot be opened\n");

return 1;

}

while ((c = getc(fp)) != EOF) {

if (c == '/') {

char next\_char = getc(fp);

if (next\_char == '/') {

// single-line comment begins

while (getc(fp) != '\n') {

comment\_chars++;

}

} else if (next\_char == '\*') {

// comment block begins

comment\_chars++;

while (1) {

c = getc(fp);

total\_chars++;

comment\_chars++;

if (c == '\*' && getc(fp) == '/') {

// comment block ends

break;

}

}

}

} else {

// not a comment

total\_chars++;

}

}

fclose(fp);

double percent = (double)comment\_chars / (double)total\_chars \* 100.0;

printf("Percentage of characters in comments: %.2f%%\n", percent);

return 0;

}

d)

#include <stdio.h>

#include <string.h>

int main() {

FILE \*fp;

char line[100];

char \*keywords[] = {"int", "float", "double", "char", "short", "long", "signed", "unsigned"};

int counts[] = {0, 0, 0, 0, 0, 0, 0, 0};

fp = fopen("input.c", "r");

if (fp == NULL) {

printf("Error: input file cannot be opened\n");

return 1;

}

while (fgets(line, 100, fp) != NULL) {

for (int i = 0; i < 8; i++) {

char \*pos = line;

while ((pos = strstr(pos, keywords[i])) != NULL) {

pos += strlen(keywords[i]);

if (\*pos == ' ' || \*pos == '\t' || \*pos == '\n') {

counts[i]++;

}

}

}

}

fclose(fp);

printf("Variables defined:\n");

printf("int: %d\n", counts[0]);

printf("float: %d\n", counts[1]);

printf("double: %d\n", counts[2]);

printf("char: %d\n", counts[3]);

printf("short: %d\n", counts[4]);

printf("long: %d\n", counts[5]);

printf("signed: %d\n", counts[6]);

printf("unsigned: %d\n", counts[7]);

return 0;

}

e)

#include <stdio.h>

int main(int argc, char \*argv[]) {

if (argc != 3) {

printf("Usage: program\_name input\_file output\_file\n");

return 1;

}

FILE \*input\_file, \*output\_file;

char c;

input\_file = fopen(argv[1], "r");

if (input\_file == NULL) {

printf("Error: input file cannot be opened\n");

return 1;

}

output\_file = fopen(argv[2], "w");

if (output\_file == NULL) {

printf("Error: output file cannot be opened\n");

return 1;

}

while ((c = getc(input\_file)) != EOF) {

putc(c, output\_file);

}

fclose(input\_file);

fclose(output\_file);

printf("File copied successfully\n");

return 0;

}

Input.c:

#include<stdio.h>

/\*This is a

comment line\*/

int main(){

char \*a="Hello World";

float f=3.14;

printf("%s %f",a,f);

}

Output:

a)

#include <stdio.h>

int main() {

FILE \*fp;

char c;

fp = fopen(\_\_FILE\_\_, "r");

do {

c = getc(fp);

putchar(c);

} while (c != EOF);

fclose(fp);

return 0;

}

b)output.c:

#include<stdio.h>

int main(){

char \*a="Hello World";

float f=3.14;

printf("%s %f",a,f);

}

c)

Percentage of characters in comments: 20.69%

d)

Variables defined:

int: 1

float: 1

double: 0

char: 1

short: 0

long: 0

signed: 0

unsigned: 0

e) $ ./a input.c output.c

File copied successfully

1. Write a program which allows you to do addition and subtraction of two integers which can be upto 40 decimal digits long (call it as huge integers). Create your own representation of huge integers and it should use as minimum space as possible.

Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_DIGITS 40

// Helper function to reverse a string

void reverse(char\* str) {

int len = strlen(str);

for (int i = 0; i < len / 2; i++) {

char tmp = str[i];

str[i] = str[len - i - 1];

str[len - i - 1] = tmp;

}

}

// Function to add two huge integers

char\* add(char\* num1, char\* num2) {

int len1 = strlen(num1);

int len2 = strlen(num2);

int max\_len = (len1 > len2) ? len1 : len2;

char\* result = (char\*) malloc(sizeof(char) \* (max\_len + 1));

int carry = 0, sum;

int i = 0, j = 0, k = 0;

while (i < len1 || j < len2) {

int digit1 = (i < len1) ? num1[i] - '0' : 0;

int digit2 = (j < len2) ? num2[j] - '0' : 0;

sum = digit1 + digit2 + carry;

carry = sum / 10;

result[k++] = (sum % 10) + '0';

i++;

j++;

}

if (carry) {

result[k++] = carry + '0';

}

result[k] = '\0';

return result;

}

// Function to subtract two huge integers

char\* subtract(char\* num1, char\* num2) {

int len1 = strlen(num1);

int len2 = strlen(num2);

int max\_len = (len1 > len2) ? len1 : len2;

char\* result = (char\*) malloc(sizeof(char) \* (max\_len + 1));

int borrow = 0, diff;

int i = len1 - 1, j = len2 - 1, k = 0;

while (i >= 0 || j >= 0) {

int digit1 = (i >= 0) ? num1[i] - '0' : 0;

int digit2 = (j >= 0) ? num2[j] - '0' : 0;

diff = digit1 - digit2 - borrow;

if (diff < 0) {

diff += 10;

borrow = 1;

} else {

borrow = 0;

}

result[k++] = diff + '0';

i--;

j--;

}

while (result[k - 1] == '0' && k > 1) {

k--;

}

result[k] = '\0';

reverse(result);

return result;

}

int main() {

char num1[MAX\_DIGITS + 1];

char num2[MAX\_DIGITS + 1];

printf("Enter the first huge integer: ");

scanf("%s", num1);

printf("Enter the second huge integer: ");

scanf("%s", num2);

// Calculate the sum and difference of the two huge integers

char\* sum = add(num1, num2);

reverse(sum);

char\* difference;

if (strcmp(num1,num2)>0){

difference = subtract(num1, num2);

}

else{

difference = subtract(num2, num1);

}

printf("The sum of the two numbers is %s\n",sum);

printf("The difference of the two numbers is %s\n",difference);

return 0;

}

Output:

Enter the first huge integer: 9999999999999999999999999999

Enter the second huge integer: 8888888888888888

The sum of the two numbers is 10000000000008888888888888887

The difference of the two numbers is 9999999999991111111111111111