# Walkthrough

1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #define STRING\_SIZE 30
3. #define RANDOM\_INTS 200
4. #include <stdio.h>
5. #include <string.h>
6. #include <time.h>
7. FILE\* makeFile(void);
8. int\* generateRandomNumber(void);
9. void writeNumbersToFile(FILE\* file, int\* numbers, int size);
10. int main(void) {
11. printf("===========================\n");
12. printf("IPC 144 Final Mock Test SLG\n");
13. printf("===========================\n");
14. FILE\* f = makeFile();
15. if (f == NULL) {
16. printf("Error creating file.\n");
17. return 1;
18. }
19. int\* randomNumbers = generateRandomNumber();
20. writeNumbersToFile(f, randomNumbers, RANDOM\_INTS);
21. fclose(f);
22. printf("File written and closed successfully.\n");
23. return 0;
24. }
25. FILE\* makeFile(void) {
26. char fileName[STRING\_SIZE + 1] = { "\0" };
27. printf("Please enter a name for your file (Max characters is 30): ");
28. scanf("%[^\n]", fileName);
29. strcat(fileName, ".txt");
30. FILE\* file = fopen(fileName, "w");
31. return file;
32. }
33. int\* generateRandomNumber(void) {
34. static int ranNumbers[RANDOM\_INTS];
35. srand(time(NULL));
36. for (int i = 0; i < RANDOM\_INTS; i++) {
37. ranNumbers[i] = rand();
38. }
39. return ranNumbers;
40. }
41. void writeNumbersToFile(FILE\* file, int\* numbers, int size) {
42. for (int i = 0; i < size; i++) {
43. fprintf(file, "%d\n", numbers[i]);
44. }
45. }

## Fill the blank

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_STUDENTS 100

#define MAX\_NAME\_LENGTH 50

typedef struct {

int id;

char name[MAX\_NAME\_LENGTH];

float grades[5];

float average;

} Student;

void inputStudent(Student\* student);

void calculateAverage(Student\* student);

void displayStudent(const Student\* student);

void updateStudent(Student\* student);

void deleteStudent(Student\* students, int\* studentCount, int id);

Student\* findStudentById(Student\* students, int studentCount, int id);

/\*

\* This program manages a list of students, including their ID, name, and grades.

\* It allows the user to add, display, update, and delete student information.

\* The program uses structures, pointers, arrays, and basic input/output functions.

\* The features include:

\* 1. Add a new student.

\* 2. Display all students.

\* 3. Update a student's information.

\* 4. Delete a student by ID.

\* 5. Find and display a student by ID.

\* 6. Exit the program.

\*/

int main() {

Student students[\_\_\_\_\_\_\_\_\_];

int studentCount = 0;

char choice;

do {

printf("\n--- Student Management System ---\n");

printf("1. Add Student\n");

printf("2. Display All Students\n");

printf("3. Update Student\n");

printf("4. Delete Student\n");

printf("5. Find Student by ID\n");

printf("6. Exit\n");

printf("Enter your choice: ");

scanf(" %c", \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

\_\_\_\_\_\_\_\_\_\_\_\_ ; // to consume the newline character left by scanf

if (choice == '1') {

if ( \_\_\_\_\_\_\_\_\_\_\_\_\_ < MAX\_STUDENTS) {

inputStudent(&students[studentCount]);

calculateAverage(&students[studentCount]);

studentCount++;

}

else {

printf("Student list is full!\n");

}

}

else if (choice == '2') {

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (&students[i]);

}

}

else if (choice == '3') {

int id;

printf("Enter student ID to update: ");

scanf("%d", &id);

Student\* student = findStudentById(students, studentCount, id);

if (student) {

updateStudent(student);

calculateAverage(student);

}

else {

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

}

}

else if (choice == '4') {

int id;

printf("Enter student ID to delete: ");

scanf("%d", &id);

deleteStudent(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

}

else if (choice == '5') {

int id;

printf("Enter student ID to find: ");

scanf("%d", &id);

Student\* student = findStudentById(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

if (student) {

displayStudent(student);

}

else {

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

}

}

else if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {

printf("Exiting the program...\n");

}

else {

printf("Invalid choice. Please try again.\n");

}

} while (choice != '6');

return 0;

}

void inputStudent(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {

printf("Enter student ID: ");

scanf("%d", \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_; // to consume the newline character left by scanf

printf("Enter student name: ");

fgets(\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, stdin);

// Remove newline character if present

int len = strlen(\_\_\_\_\_\_\_\_\_\_\_\_\_);

if (len > 0 && \_\_\_\_\_\_\_\_\_\_\_\_\_[len - 1] == '\n') {

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_[len - 1] = '\0';

}

printf("Enter 5 grades: ");

for (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) {

scanf("%f", &student->grades[i]);

}

}

void calculateAverage(Student\* student) {

float sum = 0;

for (int i = 0; i < 5; i++) {

sum += student->grades[i];

}

student->average = \_\_\_\_\_\_\_\_\_\_\_\_;

}

void displayStudent(const Student\* student) {

printf("\nStudent ID: %d\n", student->id);

printf("Student Name: %s\n", student->name);

printf("Grades: ");

for (int i = 0; i < 5; i++) {

printf("%.2f ", student->grades[i]);

}

printf("\nAverage Grade: %.2f\n", student->average);

}

void updateStudent(Student\* student) {

printf("Updating student ID: %d\n", student->id);

printf("Current name: %s\n", student->name);

printf("Enter new name (leave blank to keep current): ");

char newName[MAX\_NAME\_LENGTH];

fgets(newName, MAX\_NAME\_LENGTH, stdin);

// Remove newline character if present

int len = strlen(newName);

if (len > 0 && newName[len - 1] == '\n') {

newName[len - 1] = '\0';

}

if (strlen(newName) > 0) {

strcpy(student->name, newName);

}

printf("Enter 5 new grades (enter -1 to keep current): ");

for (int i = 0; i < 5; i++) {

float newGrade;

scanf("%f", &newGrade);

if (newGrade >= 0) {

student->grades[i] = newGrade;

}

}

}

void deleteStudent(Student\* students, int\* studentCount, int id) {

int index = -1;

for (int i = 0; i < \*studentCount; i++) {

if (students[i].id == id) {

index = i;

i = \_\_\_\_\_\_\_\_\_\_\_\_\_\_; // Exit loop

}

}

if (index != -1) {

for (int i = index; i < \*studentCount - 1; i++) {

students[i] = students[i + 1];

}

(\*studentCount)--;

printf("Student with ID %d deleted.\n", id);

}

else {

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

}

}

Student\* findStudentById(Student\* students, int studentCount, int id) {

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ // Complete the function

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

## Questions

1. What is a pointer?
2. What is a structure?
3. What are the differences between structures and arrays?
4. In header files whether functions are declared or defined?
5. What are macros? What are its advantages and disadvantages?
6. Difference between pass by reference and pass by pointer?
7. Is it possible to have negative index in an array? Why?
8. Why is it necessary to give the size of an array in an array declaration ?
9. What modular programming ?
10. What is a function ?
11. What is an argument ?
12. What are built in functions ?
13. Is it possible to have more than one main() function in a C program ?
14. Can a Structure contain a Pointer to itself? Why?
15. Difference between array and pointer ?
16. Difference between syntax vs logical error?
17. What is meant by file opening ? How we can do that?
18. Difference between an array of pointers and a pointer to an array ?
19. Differentiate between a constant pointer and pointer to a constant ?
20. Is the allocated space within a function automatically deallocated when the function returns?
21. Are the expressions \*ptr ++ and ++ \*ptr same ?