INFO1111: Computing 1A Professionalism

2023 Semester 1

Self-Learning Report

Submission number: 3

Github link: https://github.com/2245080285/info1111.git

Student name	Shuhan Chen
Student ID	sche8431
Topic	C programming
Levels already achieved	??
Levels in this report	A and B

Contents

1. Level A: Initial Understanding

1.1. Level A Demonstration

To begin learning C programming, I first set up my environment. I used Homebrew to install GCC and I used Visual Studio Code as my code editor. The first program I wrote was just printing "Hello World!" on the main function. Second step, I moved on to learn about other fundamental concepts in C programming, such as input/output operations, comments, variables, data types, and different types of loops. With this knowledge, I was able to write simple programs that could accept input from users and perform basic calculations. I also could compile programs to create an executable file. Finally, I delved into some of the more advanced features of C programming, such as memory addresses and pointers. I learned how to declare a variable as a pointer using the (*) symbol and how to obtain the memory address of a variable using the reference operator . With these concepts, I was able to manipulate memory directly and write more complex programs.

1.2. Learning Approach

As a self-learner of C programming, I first gained a basic understanding of the language's syntax and semantics by studying on various programming websites and watching video tutorials. I made sure to familiarize myself with fundamental concepts such as data types, operators, and control structures, including loops and decision statements. After obtaining a foundational understanding, I began to practice writing code by working through programming exercises and building small projects. I started with simple programs and gradually moved on to more complex ones, incorporating the use of pointers, string manipulation, and standard libraries to handle input and output. I frequently tested my code and debugged any errors that I encountered to improve my skills. .In addition, participating in online forums and programming communities was also a crucial part of my learning journey. I asked questions, shared ideas, and received feedback from experienced programmers. Reading other people's code helped me to understand different coding styles and techniques and improve my own code.

1.3. Challenges and Difficulties

As a self-learner of C programming, I found that some of the most challenging aspects were understanding the use of pointers and getting used to the syntax of the language. Pointers are a fundamental part of the language, but they can be quite challenging to understand, especially for beginners. It took me a significant amount of time to study and practice using pointers to fully grasp their concept. Another difficulty I encountered was getting accustomed to the syntax of the language, which is quite different from other programming languages like python. It took me some time to get used to the syntax of C, including its use of semicolons, curly braces, and parentheses. One of the most challenging topics I encountered while learning C programming was understanding the concept of structures. Structures allow you to group together related data elements under a single user-defined data type. While structures provide a powerful tool for organizing complex data, they can also be difficult to understand, especially for beginners. I struggled to understand how to declare and initialize structures, how to access individual structure members, and how to pass structures to functions.

1.4. Learning Sources

https://www.w3schools.com/c/index.php [1]	well-structured tutorial including several topics and examples of C programming.The tutorial is divided into various sections, including an introduction to C, basic concepts, data types, operators, control structures, functions, pointers, and more. I use it to study the fundamental concepts about C
https://www.geeksforgeeks.org/c-programming-language/ [2]	The platform also offers a vast library of practice problems and real-world applications. I used it to help sharpen my skills and gain a deeper understanding of C programming.By engaging with these resources, I not only refined my skill set but also gained a comprehensive understanding of the various aspects of C programming.
`https://www.youtube.com/watch?v=rLf3jnHxSmU&list=PLBlnK6fEyqRggZZgYpPMUxdY1CYkZtARR` [3]	C programming tutorials for beginners.Videos combine visual and auditory elements, making it easier for me with different learning styles to grasp complex concepts. Seeing code being written in real-time, along with verbal explanations. There are many step-by-step demonstrations of programming concepts that help me understand them quickly.

1.5. Application artifacts

I have developed a Student Record Database application in the C programming language. The program utilizes standard libraries such as `<stdio.h>`, `<string.h>`, and `<stdlib.h>`. It manages student records, comprising names, ages, genders, ID numbers, and math, science, and English grades.

The "student" struct is used to store individual student records:

```
struct student {
    char name[50];
    int age;
    char gender[10];
    int id_number;
    float grade_math;
    float grade_science;
    float grade_english;
};
```

A pointer to the memory of these structs, named "records", facilitates access to student records. The program also enforces a maximum limit of 100 students.

A pointer to the memory of these structs, named "records" (`struct student *records;`), facilitates access to student records. The program also enforces a maximum limit of 100 students, set using `#define MAX_STUDENTS 100`.

The database is supported by several functions:

- `add_student()`: Adds a student by printing instructions to assist the user in providing student details. It validates the input using the `check_student_detail` function and stores it in the records if valid.

```
void add_student() {
    ...
    if (check_student_detail(new_student) == 1){
        records[num_records] = new_student;
    }else{
        printf("invalid detail of student ");
    }
    (num_records)++;
}
```

- `search_student(int id_number)`: Searches for a student by ID and displays their details if found; otherwise, it informs the user that the student was not found.

```
void search_student(int id_number) {
    ...
    for (i = 0; i < num_records; i++) {
        if (records[i].id_number == id_number) {
            printf("Name: %s\n", records[i].name);
            ...
            return;
        }
    }
    printf("No student with ID number %d found\n", id_number);
}
```

- `delete_student(int id_number)`: Deletes a student's record from the database using their ID. If found, the student's record is removed from the records array, and the remaining records are adjusted accordingly. Messages are printed to the user based on the outcome.

```
void delete_student(int id_number) {
    ...
    for (i = 0; i < num_records; i++) {
        if (records[i].id_number == id_number) {
            ...
            (num_records)--;
            printf("Student with ID number %d deleted\n", id_number);
            return;
        }
    }
    printf("No student with ID number %d found\n", id_number);
}
```

- `display_all_students()`: Iterates through the records array with a for loop and displays information about all students in the database.

```
void display_all_students() {
```

```
    ...
    for (i = 0; i < num_records; i++) {
        printf("Name: %s\n", records[i].name);
        ...
        printf("\n");
    }
}
```

- Quit: Breaks the loop to exit the continuously displayed menu.

The program's main function uses a while loop to display the menu, take user input, and execute the corresponding functions. Dynamic memory allocation with `malloc(MAX_STUDENTS * sizeof(struct student))` is employed to allocate memory for storing records, considering the size of the student struct multiplied by the maximum limit. Lastly, the `free()` function is used to release the allocated memory before exiting.

2. Level B: Basic Application

2.1. Level B Demonstration

I make a student records management system, this project allows me to add, search, delete, and display student records using the C programming language. I have implemented a struct to represent the student records with fields such as name, age, gender, ID number, and grades for math, science, and English and us malloc to set memory to store students records.

2.2. Application artifacts

I have developed a Student Record Database application in the C programming language. The program utilizes standard libraries such as `<stdio.h>`, `<string.h>`, and `<stdlib.h>`. It manages student records, comprising names, ages, genders, ID numbers, and math, science, and English grades.

The "student" struct is used to store individual student records:

```
struct student {
    char name[50];
    int age;
    char gender[10];
    int id_number;
    float grade_math;
    float grade_science;
    float grade_english;
};
```

A pointer to the memory of these structs, named "records", facilitates access to student records. The program also enforces a maximum limit of 100 students.

A pointer to the memory of these structs, named "records" (`struct student *records;`), facilitates access to student records. The program also enforces a maximum limit of 100 students, set using `#define MAX_STUDENTS 100`.

The database is supported by several functions:

- `add_student()`: Adds a student by printing instructions to assist the user in providing student details. It validates the input using the `check_student_detail` function and stores it in the records if valid.

  ```
  void add_student() {
      ...
      if (check_student_detail(new_student) == 1){
          records[num_records] = new_student;
      }else{
          printf("invalid detail of student ");
      }
      (num_records)++;
  }
  ```

- `search_student(int id_number)`: Searches for a student by ID and displays their details if found; otherwise, it informs the user that the student was not found.

  ```
  void search_student(int id_number) {
      ...
      for (i = 0; i < num_records; i++) {
          if (records[i].id_number == id_number) {
  ```

```
            printf("Name: %s\n", records[i].name);
            ...
            return;
        }
    }
    printf("No student with ID number %d found\n", id_number);
}
```

- `delete_student(int id_number)`: Deletes a student's record from the database using their ID. If found, the student's record is removed from the records array, and the remaining records are adjusted accordingly. Messages are printed to the user based on the outcome.

```
void delete_student(int id_number) {
    ...
    for (i = 0; i < num_records; i++) {
        if (records[i].id_number == id_number) {
            ...
            (num_records)--;
            printf("Student with ID number %d deleted\n", id_number);
            return;
        }
    }
    printf("No student with ID number %d found\n", id_number);
}
```

- `display_all_students()`: Iterates through the records array with a for loop and displays information about all students in the database.

```
void display_all_students() {
    ...
    for (i = 0; i < num_records; i++) {
        printf("Name: %s\n", records[i].name);
        ...
        printf("\n");
    }
}
```

- Quit: Breaks the loop to exit the continuously displayed menu.

The program's main function uses a while loop to display the menu, take user input, and execute the corresponding functions. Dynamic memory allocation with `malloc(MAX_STUDENTS * sizeof(struct student))` is employed to allocate memory for storing records, considering the size of the student struct multiplied by the maximum limit. Lastly, the `free()` function is used to release the allocated memory before exiting.

3. Level C: Deeper Understanding

Level C focuses on showing that you have actually understood the tool or technology at a relatively advanced level. You will need to compare it to alternatives, identifying key strengths and weaknesses, and the areas where this tool is most effective.

3.1. Strengths

What are the key strengths of the item you have learnt? (50-100 words)

3.2. Weaknesses

What are the key weaknesses of the item you have learnt? (50-100 words)

3.3. Usefulness

Describe one scenario under which you believe the topic you have learnt could be useful. (50-100 words)

3.4. Key Question 1

Note: This question is in the table in the 'Self Learning: List of Topics' page on Canvas. (50-100 words)

3.5. Key Question 2

Note: This question is in the table in the 'Self Learning: List of Topics' page on Canvas. (50-100 words)

4. Level D: Evolution of skills

4.1. Level D Demonstration

This is a short description of the application that you have developed. (50-100 words). **IMPORTANT:** *You might wish to submit this as part of an earlier submission in order to obtain feedback as to whether this is likely to be acceptable for level D.*

4.2. Application artifacts

Include here a description of what you actually created (what does it do? How does it work? How did you create it?). Include any code or other related artefacts that you created (these should also be included in your github repository).

If you do include screengrabs to show what you have done then these should be annotated to explain what it is showing and what the application does.

4.3. Alternative tools/technologies

Identify 2 alternative tools/technologies that can be used instead of the one you studied for your topic. (e.g. if your topic was Python, then you might identify Java and Golang)

4.4. Comparative Analysis

Describe situations in which both your topic and each of the identified alternatives would be preferred over the others (100-200 words).

Bibliography

[1] "C programming tutorial." Well-structured tutorial including several topics and examples of C programming. Used for studying fundamental concepts about C. (), [Online]. Available: https://www.w3schools.com/c/index.php.

[2] "C programming language." Tutorial with more detailed and deeper explanation. (), [Online]. Available: https://www.geeksforgeeks.org/c-programming-language/.

[3] "C programming tutorials." Several C programming tutorials for beginners, helping to understand C topics more easily. (), [Online]. Available: https://www.youtube.com/watch?v=rLf3jnHxSmU&list=PLBlnK6fEyqRggZZgYpPMUxdY1CYkZtARR.