CS1412, Spring 2021 Problem Solving & Program Design in C

Lab 7 - In Lab Assignment

Due end of the lab session

**Name: Bibek Dhungana**

Acknowledge your collaborators or source of solutions, if any. **Submission by the end of the LAB is required.** Please type your answers, handwritten submission will not be accepted. Do all of the following. A subset of your solutions will be graded.

1. How a struct could be used to show which field in a union is being used?

When many variables are associated in the union, the complier allocate the memory for size of largest member of union. So, memory allocated is shared by each member of union. That is why only one member can be accessed at single time. Generally, union do not track which field is being used. But, if we want to track which field is being used, we can define union inside the struct to track it. Also, enumerated type flag can also be used in unions.For eg:

// declaring the union inside struct to track it.

struct Student

{

char name[10];

int age;

int rollNo;

union Grade

{

int marks;

char letterGrade;

} myUnion;

};

//declaring the structure student 1

struct Student Student1;

student.myUnion.letterGrade = ‘A’

Now, we can use if condition to see what field is being used.

void \*ptr = &student1.myUnion

if(\*((char\*)ptr) == 'A')

printf(“ use char variable”);

else

printf(“use int variable”);

similarly, we can use similar format to track the field being used for other union too.

void \*ptr = &student1.myUnion

1. Members of a union are accessed as .
   1. union-name.member
   2. union-pointer->member
   3. Both a & b
   4. None of the mentioned

Answer is C.

For eg, we have a union

union Student {

int num1;

float num2;

char name[20];

};

Case 1:

Union Student student1;

Student1.num1 = 10;

Case 2:

Union myPointer \*myPointer;

myPointer-> num1 = 10;

1. Write a program where you declare and initialize an array of 10 student\_t structures and write a code segment that displays on separate lines the names (last name, first name) and ID of all the students in the list.

**CODE**

//including all the required libraries

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

//declaring the structure student\_t

//we can also use typedef in struct.so that struct must not be used in every //declaration.

struct student\_t {

char firstName[20];

char lastName[20];

int studentId;

};

int main(void) {

struct student\_t students[10];

//initializing the student 1

strcpy(students[0].firstName, "Bibek");

strcpy(students[0].lastName, "Dhungana");

students[0].studentId = 100;

//initializing the student 2

strcpy(students[1].firstName, "Ram");

strcpy(students[1].lastName, "Karki");

students[1].studentId = 101;

//initializing the student 3

strcpy(students[2].firstName, "Samrat");

strcpy(students[2].lastName, "Pokhrel");

students[2].studentId = 102;

//initializing the student 4

strcpy(students[3].firstName, "Ni");

strcpy(students[3].lastName, "Sultana");

students[3].studentId = 103;

//initializing the student 5

strcpy(students[4].firstName, "Ryan");

strcpy(students[4].lastName, "Bain");

students[4].studentId = 104;

//initializing the student 6

strcpy(students[5].firstName, "Ni");

strcpy(students[5].lastName, "Sultana");

students[5].studentId = 105;

//initializing the student 7

strcpy(students[6].firstName, "Mark");

strcpy(students[6].lastName, "Zukerberg");

students[6].studentId = 106;

//initializing the student 8

strcpy(students[7].firstName, "Elon");

strcpy(students[7].lastName, "Musk");

students[7].studentId = 107;

//initializing the student 9

strcpy(students[8].firstName, "Jeff");

strcpy(students[8].lastName, "Bezos");

students[8].studentId = 108;

//initializing the student 10

strcpy(students[9].firstName, "Steve");

strcpy(students[9].lastName, "Jobs");

students[9].studentId = 109;

//printing down the information

printf("\nThe information of student in format:\n");

printf("(lastName,FirstName)\n student ID\n");

//printing down the student information

for (int i = 0; i < 10; i++) {

printf("\n\nStudent %d:\n", i + 1);

printf("(lastName,firstName):%s,", students[i].lastName);

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

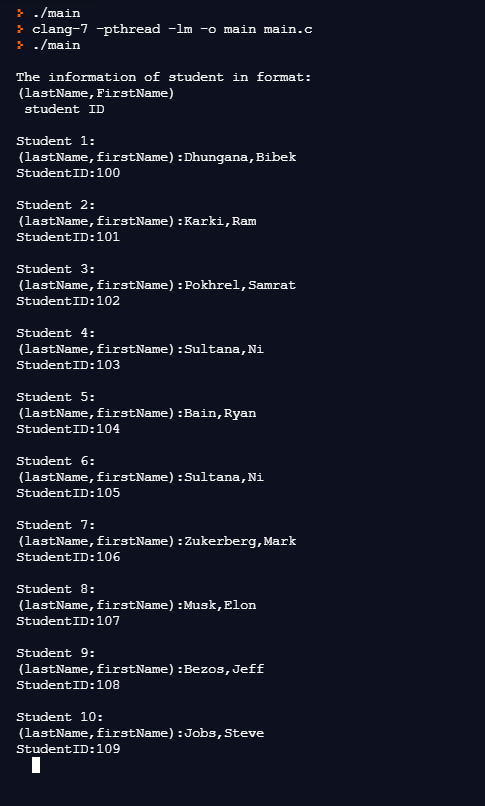
printf("StudentID:%d\n", students[i].studentId);

}

return 0;

}

**OUTPUT**



We can also make this code shorter by taking input from the user dynamically. This will make the code shorter by using scanf inside the loop to initialize array of structure.

Code:

//including all the required libraries

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

//declaring the structure student\_t

struct student\_t {

  char firstName[20];

  char lastName[20];

  char studentId[20];

};

int main(void) {

  //creating array of structure that can hold 10 student

  struct student\_t students[10];

  //taking input from the user about the student

  for (int i = 0; i < 10; i++) {

    printf("\nEnter the student %d firstName:", i + 1);

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

    printf("Enter the student %d lastName:", i + 1);

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

    printf("Enter the student %d ID:", i + 1);

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

  }

  //printing down the information

  printf("\nThe information of student in format:\n");

  printf("(lastName,FirstName)\n student ID\n");

  //printing down the student information

  for (int i = 0; i <10; i++) {

    printf("\nStudent %d:\n",i + 1);

    printf("Name:%s", students[i].lastName);

    printf(",");

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

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

  }

  return 0;

}