ELEC 278

Fundamentals of Information Structures

Overview of C

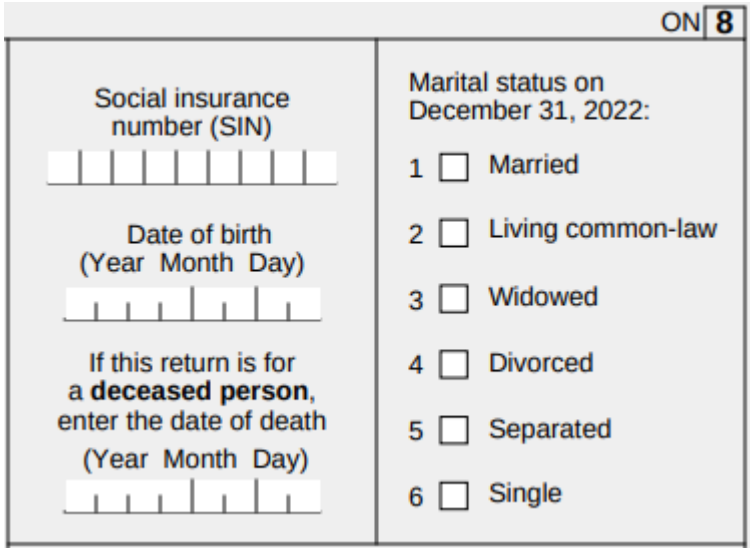
Dr. Jianbing Ni

Department of Electrical & Computer Engineering

Queen's University at Kingston

Data Structure

- Data structure is a way to store and organize data in order to facilitate access and modification.
- How can we represent the *completed* contents of this form section in computer memory?



Variables in C

We declare a variable by writing its type followed by its name:

- int total; // An integer variable named 'total’
- char first_initial; // A character variable
- int numbers[10]; // An array of 10 numbers

Variables in C

- Variables are **containers** for storing data values, like numbers and characters.
- A user-defined or a user-readable custom name assigned to a memory location
- **A name of the memory location**.

Address	Values
0xFF000000	total
0xFF000002	first_initial
oxFF000003	number[0]
0xFF000005	number[1]
…	…

Variables in C

- Computer memory consists of many boxes (memory cell), each of which can store some amount of information (usually 1 byte; 256 possible values).
- A program can read or write the contents of any box using its number (**memory address**).
- A variable is just a container that we refer to by some label; that label is just a placeholder for its variable.

Address	Values
0xFF000000	total
0xFF000002	first_initial
oxFF000003	number[0]
0xFF000005	number[1]
…	…

Primitive Data Types in C

The type of the variable describes how the contents of the box(es) referred to by the variable should be interpreted.

- **int** – holds typical integer (at least 16 bits)
- **char** – holds one character (byte)
- **float** – floating point number – at least 32 bits
- **double** – more precise floating point – usually 64 bits

Declaring Variables

```
int    j, k, m;
char   p, q, r;
float  f;
short int   a = 5;
char b = 'Z';
float pi = 3.14159;
double abigname99 = -3.7e-4
```

Strings

- A string is a sequence of characters, identified by double quote marks:

  char str1[12] =“hello”;

- This really means 6 characters in a row:
  ‘h’ ‘e’ ‘l’ ‘l’ ‘o’ ‘\0’
- Strings in C are stored in arrays of characters.

```
char str1[12] = "Hello";
char str2[12] = "World";
char str3[12];
strcpy(str3, str1);
strcat( str1, str2);
```

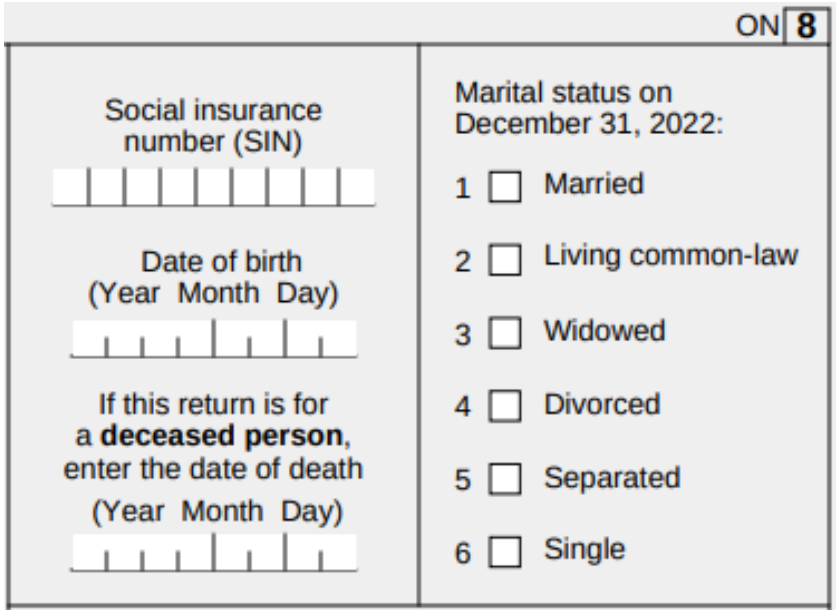
Arrays

- Arrays are sets of identical data items:
  int g[100] – 100 integers, g[0] to g[99]
  char s[50] – 50 chars, s[0] to s[49]

  int sin[9];

  int date_of_birth[8];

  int date_of_death[8];



Struct

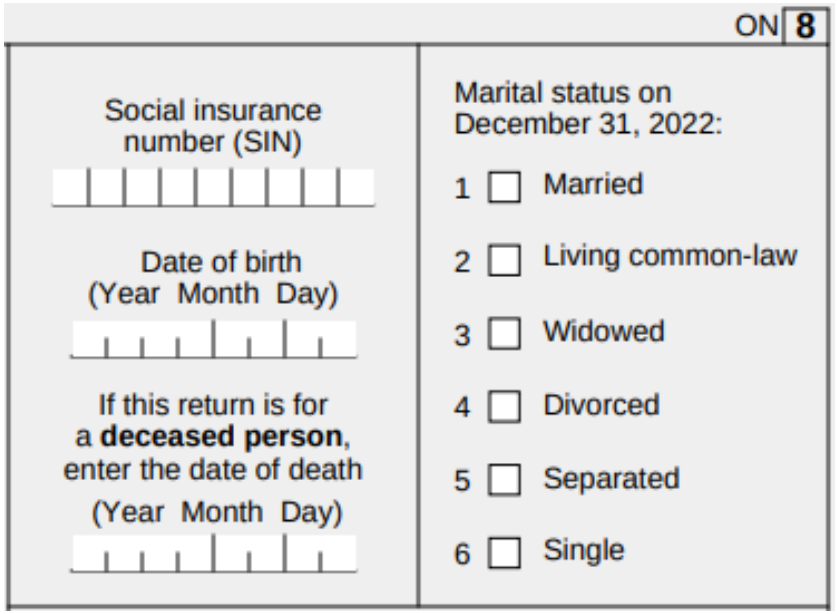
- A user-defined data type
- used to group items of possibly different types into a single type.
- Syntax of struct:

```
struct structure_name {
    data_type    member_name1;
    data_type    member_name2;
    ....
};
```

The items in the structure are called its **member**

Struct

```
struct tax_info {
        int sin[9];
        int date_of_birth[8];
        int date_of_death[8];
        int marital_status;
};
```



Struct

- **Declare struct Variables**

Similar to:
int a, b;

Method 1: If structure_name has been created,

struct structure_name variable_name;

struct structure_name variable_name1, variable_name2;

Method 2: Create structure_name and declare variables.

```
struct structure_name {
        data_type    member_name1;
        data_type    member_name2;
           ....
}variables;
```

Struct

- **Declare struct Variables**

```
struct tax_info my_tax_info;
struct tax_info Alice_tax_info, Bob_tax_info;
struct tax_info {
        int sin[9];
        int date_of_birth[8];
        int date_of_death[8];
        int marital_status;
} Alice_tax_info, Bob_tax_info, Carlos_family_tax[5];
% Carlos_family_tax[5] is a struct tax_info array of size 5.
```

Struct

Assign values to structure members

- Curly brackets '{}': initialize all the members

Struct tax_info Evetax= {{1,2,3,4,5,6,7,8,9},
{1,9,9,8,0,3,1,3},{2,0,2,2,1,1,1,4}, 5};

```
struct Student{
    char name[50];
    int class;
    char section;
};
struct Student student1={"Alice" ,1,'A'};
```

Struct

Access structure members

- Dot '.' : Member operator

  Alice_tax_info.sin[9]={1,2,3,4,5,6,7,8,9};

  Bob_tax_info. marital_status=4;

```
struct Student {
    char name[50];
    int class;
    char section;
}student1;

strcpy(student1.name,"Alice");
student1.class = 1;
student1.section = 'A';
```

Struct

```
#include <stdio.h>
#include <string.h>
struct student
{       int id;
        char name[20];
        float percentage;
};
int main()
{
        struct student record;
        record.id=1;
        strcpy(record.name, "Raju");
        record.percentage = 86.5;
        printf(" Id is: %d \n", record.id);
        printf(" Name is: %s \n", record.name);
        printf(" Percentage is: %f \n", record.percentage);
        return 0;
}
```

Define the data type

Similar to:
int a, b;

Declare the structure variable

Assign values to structure members

Access structure members

Struct and Typedef

Typedef: To create an alias name for data types.

Used with structures to simplify the syntax of declaring variables.

```
struct Student {
     char name[50];
     int class;
     char section;
};
Main(){
   Struct Student s1,s2;
}
```

```
Typedef struct Student {
     char name[50];
     int class;
     char sect
}Students;
Main(){
   students s1,s2;
}
```

Same as:
int a, b;

Pay attention to differences!

```
struct Student {
     char name[50];
     int class;
     char section;
}student1;
```

Struct and Typedef

```
#include <stdio.h>
#include <string.h>
typedef struct student
{       int id;
        char name[20];
        float percentage;
}Students;
int main()
{
        Students record;
        record.id=1;
        strcpy(record.name, "Raju");
        record.percentage = 86.5;
        printf(" Id is: %d \n", record.id);
        printf(" Name is: %s \n", record.name);
        printf(" Percentage is: %f \n", record.percentage);
        return 0;
}
```

Create an alias for structure student

Declare structure variable

Nested Structures

Create structures within a structure

```
struct complex {
      int imag;
      float real;
};
struct number {
      struct complex comp;
      int integers;
} num1, num2;
```

To assign values to members:

```
num2.comp.imag = 11;
num2.integers=22;
```

Struct and Pointers

- Create pointers to struct: struct followed by structure name to which the pointer will point to followed by pointer name.

```
struct structure_name *structure_pointer;
```

- To initialize a structure variable, provide the address of the structure variable

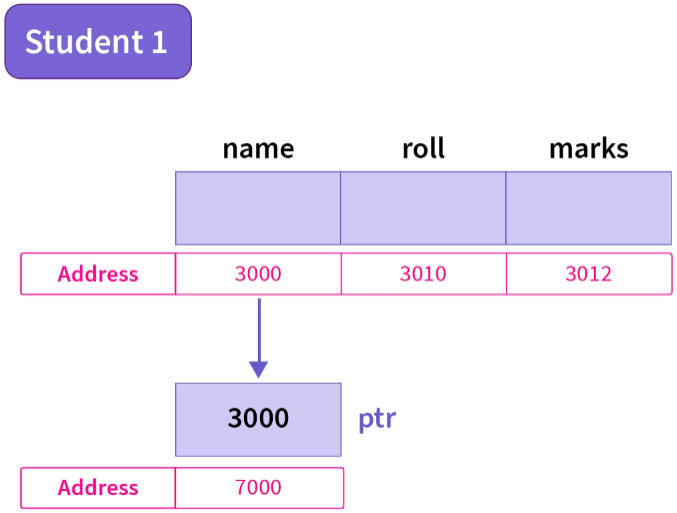
```
structure_pointer = &structure_variable;
```

The structure pointer can be initialized during declaration.

```
struct structure_type *structure_pointer = &structure_variable;
```

Struct and Pointers

```
struct student{
   char name[10];
   int roll;
   int marks;
};
int main() {
   struct student Student1;
   struct student *ptr;
   ptr=&Student1;
}
```



Struct and Pointers

Access the Structure Member Using Pointer

- 1. Using asterisk (*) and dot (.) operator

```
#include<stdio.h>
struct Coordinate {
    // declare structure members
    int x,y;
};
int main() {
    struct Coordinate first_point;
    struct Coordinate *cp;
    cp = &first_point;
    (*cp).x = 5;
    (*cp).y = 10;
    printf("First coordinate (x, y) = (%d, %d)", (*cp).x, (*cp).y);
    return 0;
}
```

- cp is a pointer that points to the structure variable first_point
- *cp and first_point are functionally identical

Struct and Pointers

Access the Structure Member Using Pointer

- 2. Using the arrow operator: pointer_name->member_name

```
#include<stdio.h>
struct Student {
    char name[30];
    int age;
    int roll_number;
};
int main() {
    struct Student student_1;
    struct Student *sp = &student_1;
    scanf ("%s", sp->name);
    scanf ("%d", &sp->age);
    scanf ("%d", &sp->roll_number);
    printf ("\tName: %s\n", sp->name);
    printf ("\tAge: %d\n", sp->age);
    printf ("\tRoll Number: %d", sp->roll_number);
    return 0;}
```

- sp is a pointer that points to the structure variable student_1
- xp-> members zaccess the structure member

Struct and Function

- Method 1: Pass the structure to a function by address.
- The structure can be accessed from called function by its address.

```
#include <stdio.h>
struct student {
    char name[50];
    int roll;
    float marks;
};
void display(struct student* st_obj)
{
    printf("Name: %s\n", st_obj->name);
    printf("Roll: %d\n", st_obj->roll);
    printf("Marks: %f\n", st_obj->marks);
}
int main()
{
    struct student st1
    display(&st1);
    return 0;
}
```

struct student* st_obj=&st1;

Function Call By Reference

Struct and Function

- Method 2: Pass the structure to a function by value.
- The structure can be accessed from called function by its value.

```
#include <stdio.h>
struct student {
    char name[50];
    int roll;
    float marks;
};
void display(struct student  st_obj)
{
    printf("Name: %s\n", st_obj.name);
    printf("Roll: %d\n", st_obj.roll);
    printf("Marks: %f\n", st_obj.marks);
}
int main()
{
    struct student st1
    display(st1);
    return 0;
}
```

Function Call By Value