Programming Fundamentals (CS-302)

(Structures)

Dr. Ihsan Ullah

Lecturer
Department of Computer Science & IT
University of Balochistan

Introduction

- A variable can store one element at a time
- An array can store a collection of elements with the same type
- How to handle a collection of mixed types of data? Such as Grades and marks
- C language provides structures for a collection of elements with different types
- Structure can be considered as a user-defined data type

Structure declaration

- A structure is declared through the keyword struct
- The general form is

```
struct <structure name>
{
  structure element 1 ;
  structure element 2 ;
   ......
}
```

- Declaration of a structure does not reserve any memory
- Memory is reserved when a structure variable is defined

Example

- A structure storing marks and grade of a student:

```
struct result
{
int rno;
int marks;
char grade;
} ;
```

- To define variables of type structure result:
  - struct result s1, s2;
- To access elements of structure variable s1:
  - s1.rno, s1.marks, s1.grade
- See example struct1.c, strucStr.c
- Modify struct1.c to take input from the user

Data types

- Integers
  - short, int, long
  - On a 32 bit machine occupy 2,4 and 4 bytes respectively
  - On a 16-bit machine occupy 2,2 and 4 bytes respectively
  - Adding unsigned with all these types increases its storage limit allowing only positive integers
  - The integer range in a two bytes space is -32768 to +32767. Using unsigned integer allows to store values in range from 0 to 65535
- Characters
  - To store character type char is used occupying one byte
- Fractions
  - Float, double & long double occupying 4, 8 and 10 bytes respectively

Variable storage classes

- A variable defined in C language refers to some physical location within the computer
- Such locations are memory and CPU registers
- A storage class determines
  - The type of location (memory/register)
  - Default initial value
  - The scope of a variable (visibility in functions)
  - Life of variable

Variable storage classes

- Four storage classes in C
  - Automatic
  - Register
  - Static
  - External

Automatic storage class

- Storage location is memory
- Its default initial value is unpredictable (garbage)
- Its scope is within the block in which it is defined
- Life of automatic variable remains until the control remains within the same block
- Syntax auto int a; //auto is optional
- Keyword auto is rarely used since it is the default type

Register storage class

- Storage location is CPU registers
- Default initial value is garbage
- Scope is local to the block in which variable is defined
- life is until the control remains within the block
- Faster access than the variable stored in memory
- Syntax: `register int a;`
- Processed as auto, if free register is not available

Static storage class

- Static variable is stored in memory
- Its default initial value is 0
- Its scope is local to the block in which it is defined
- Life is until the control remains in the program (holds its value during function calls)
- Syntax: `static int a;`
- See example `staticVar.c`

External storage class (global)

- Called global variables
- Storage location is in memory
- Its scope is global to the program
- Life remains throughout the program
- Defined above all functions and remains visible to all of them
- Default initial value is zero
- See example globalVar.c

Name input

- C language does not have a string type
- Strings are stored in character arrays
- To input a string %s is used as a format specifier
- & sign is not required in the scanf() function since the name of array carries its first address
- See example resultString.c, resultStringInput.c
- To copy a string into another strcpy() function is used
- To check the equality of two strings strcmp() is used

Array of structures

- Just like arrays of int and char, array of structures can be created too
- An array of structure is defined as struct stuctureName varname[array length];
- To access elements of an array of structures, varname[index number].field
- For a complete example see arrStdResult.c