ESc101: Structures

Instructor: Krithika Venkataramani
Semester 2, 2011-2012

- Storing records may require grouping of different type of information per unit
- E.g. ESc101 Marks Record of a student has
  - name of student: string
  - roll number: integer/string
  - multiple quiz marks: float
  - multiple lab exam marks: float
  - mid-sem marks: float
  - pro-rate records: string + integer
- Structures can be used to group different types of variables
- Structure members: the different variables
Structure declaration

- A structure is declared using the keyword `struct`.

```c
struct record
{
    float quiz1;
    char name[30];
};
```

- The structure, `record`, has 2 members, `quiz1` and `name`.

- A variable of a particular structure type can be defined using `struct ESc101Student record;`.

- Explicit definition can be done through `typedef`.

```c
typedef struct record ESc;
```

- The structure, `record`, can now be defined as `ESc Student1;`.

Initialization and Member Access

- Structures can be initialized during declaration.

```c
ESc Student1 = {8.0 , "Dilip"};
```

- By default, they are initialized to 0 (or `\0`).

- Its members can be explicitly assigned values.

- `. notation to access members`.

```c
structure_variable.member name
```

```c
Student1.quiz1 = 8.0;
strcpy(Student1.name,"Dilip");
```

- Members behave just like ordinary variables.

- Size of a structure is the combined size of its members.

- Example: Size of `ESc` is $4 + 30 = 34$ bytes.
Functions using structures

- Since structures are variables, a function can return them

```c
typedef struct Point{
  float x;
  float y;
} point;

point copy_point ( point s)
{
  point p;
  p.x = s.x;
  p.y = s.y;
  return p;
}
```

```c
void main()
{
  point p1 = {9.0, 4.0}, p2;
  p2 = copy_point(p1);   //p2 has same member values as p1
}
```

- Functions can be used to create structures

```c
point create_point(float x, float y)
{
  point p;
  p.x = x;
  p.y = y;
  return p;
}
```

```c
q = create_point(9.0, -3.0);
```

- Copying can also be done simply by

```c
q = p;
```

- Structures cannot be compared

```c
if (q == p)    // error
```

Krithika Venkataramani (krithika@cse.iitk.ac.in)
### Pointers to Structures

- A pointer to a structure can be defined
  
  ```
  point *ptr, p;
  ptr = &p;
  ```

- > notation to access members using pointers

- structure pointer->member name

- ptr->x is same as (*ptr).x

- When a pointer to structure is passed to a function, modifying the elements of the structure inside the function becomes permanent (reflected outside the function)

```c
void modify (point *p, double c, double d)
{
    p->x = c;
    p->y = d;
}
```

### Operations on structures

```c
#include<stdio.h>
#include<string.h>

struct record
{
    float quiz1;
    char name[30];
}; // defining a structure

typedef struct record ESc; // defining a new type using structure

ESc new_student(float marks, char StudentName[]) // structure as return value
{
    ESc Student1;
    Student1.quiz1 = marks;
    strcpy(Student1.name, StudentName);
    return Student1;
}
```
Modifying members using functions

void modify_wrong(ESc s, float marks, char StudentName[])
{
    s.quiz1 = marks; // modifying members inside function is temporary
    strcpy(s.name, StudentName);
}
void modify_pointer(ESc *p, float marks, char StudentName[])
{
    p->quiz1 = marks; /* modifying members using structure pointer is
permanent*/
    strcpy(p->name, StudentName); // -> notation
}

Operations on structures (cont.)

int main()
{
    struct record s1, s2; // declaring using structure
    ESc student1; // declaring using type
    ESc student2 = {8.5, "Aditya"}; // initializing during declaration
    float d;
    ESc *ptr;
    printf("s1: %f %s\n", s1.quiz1, s1.name); // by default, values are 0
    printf("student1: %f %s\n", student1.quiz1, student1.name); // by default, values are 0
    printf("student2: %f %s\n", student2.quiz1, student2.name);
    s2.quiz1 = 4.0; // accessing or modifying the members in a structure
    strcpy(s2.name,"Bharat"); // . notation
    printf("s2 assigned through . operator: %f %s\n", s2.quiz1, s2.name);
}
Operations on structures (cont.)

```c
//s1 = {9.0, "Chaitanya");   // error
//printf("%f %s
", s1.quiz1, s1.name);
s1 = new_student(7.0, "Dilip");
printf("s1 returned through function: %f %s\n", s1.quiz1, s1.name);
modify_wrong(s2, 6.0, "Gagan");
printf("s2 modified in function: %f %s\n", s2.quiz1, s2.name);
ptr = &s1;
modify_pointer(ptr, 2.0, "Hari");
printf("s1 modified through pointer: %f %s\n", s1.quiz1, s1.name,
ptr->quiz1, ptr->name);
```

//if (s1 == s2) // error
//printf("Equal structures\n");
```

Program output: Operations on structures

- s1: -0.000000
- student1: 0.000000
- student2: 8.500000 Aditya
- s2 assigned through . operator: 4.000000 Bharat
- s1 returned through function: 7.000000 Dilip
- s2 modified in function: 4.000000 Bharat
- s1 modified through pointer: 2.000000 Hari      2.000000 Hari

Krithika Venkataramani (krithika@cse.iitk.ac.in)
Pointer in a Structure

- A structure can have a pointer as its member

```c
typedef struct record
{
    float quiz1;
    char * name;
} ESc;
```

- Declaring a variable of type `ESc` just declares the pointer `name`: it does not allocate space for it

```c
ESc s;
strcpy(s.name, "Dilip"); // error as no space is allotted to s.name
```

- Memory for `name` has to be allocated explicitly using `malloc`

```c
s.name = (char *)malloc(30 * sizeof(char));
strcpy(s.name, "Dilip");
```

Pointers as members

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct record
{
    float quiz1;
    char * name;
} ESc;

void main()
{
    ESc s;
    s.name = (char *)malloc(30 * sizeof(char)); // Need to assign space first
    scanf("%f%s", &s.quiz1, s.name);
    printf("%f %s\n", s.quiz1, s.name);
    strcat(s.name, " A.");
    printf("%f %s\n", s.quiz1, s.name);
    free(s.name); // important to free space
}
```
Nested Structures

- A structure can have another structure as its member

```c
typedef struct twoD
{
    float x;
    float y;
} point;

typedef struct Line
{
    point p;
    point q;
} line;
```

- Value x of point p of variable line1 of type line can be accessed as: `line1.p.x`

- The . operator has left-to-right associativity

Array of Structures

- An array of structures can be simply defined as

```c
point t[3];
```

- Each individual structure is accessed as `t[0]`, etc.

- A member of a structure is accessed as `t[i].x`, etc.

- All operations allowed on normal arrays are allowed on array of structures
Array of Structures

```c
#include <stdio.h>

typedef struct twoD
{
    double x;
    double y;
} point;

void main()
{
    point t[3];
    int i;
    for (i = 0; i < 3; i++)
    {
        t[i].x = i;
        t[i].y = 2 * i;
        printf("%lf %lf
", t[i].x, t[i].y);
    }
}
```

The content of some of these slides are from the lecture slides of Prof. Arnab Bhattacharya

Karthika Venkataramani (krithika@cse.iitk.ac.in)