Programming Languages

Bibhas Adhikari

IIT Kharagpur

October 29, 2020

Advanced data types/Data structures

Structure

- These are a collection of variables related in real life, but not necessarily in data type
- Structures are most commonly used to define an object—a person, a place, a thing—or similarly a record in a database or file
- Structures use new keywords to build a well-defined collection of variables
- A structure can be created by building the structure definition using the *struct* keyword followed by braces, with individual variables defined as members. For instance,

```
struct math {
int x;
int y;
int result;
};
```

Advanced data types/Data structures

- The above program statements create a structure definition called *math* which contains three integer-type members
- The keyword *math* is also known as structure tag that is used to create instances of the structure
- However, when structure definitions are created using the *struct* keyword, memory is not immediately allocated for the structure until an instance of the structure is created such as
  struct math aProblem;
- In the above program, an instance of structure, a variable *aProblem* is created which is a variable of structure type
- A structure can also be initialized the same way as an array is initialized, such as
  struct math aProblem $= \{0, 0, 0\}$;

Advanced data types/Data structures

- Once an instance of the structure is created, the members of the structure can be accessed via the dot operator (.), also known as dot notation described as follows:
  $aProblem.x = 10;$
  $aProblem.y = 10;$
  $aProblem.result = 20;$
  $printf(\text{“}\backslash n\%d$ plus $\%d\text{”}$, aProblem.x, aProblem.y);
  $printf(\text{“}$ equals $\%d\backslash n\text{”}$, aProblem.result);

Advanced data types/Data structures

However, members of structures need not have the same data type, as shown in the following program:

```
struct employee {
char fname[10];
char lname[10];
int id;
float salary;
};
main( )
{
//create instance of employee structure
struct employee emp1;
```

Advanced data types/Data structures

```
//assign values to members
strcpy(emp1.fname, "Bibhas");
strcpy(emp1.lname, "Adhikari");
emp1.id = 123;
emp1.salary = 50000.00;
//print member contents
printf("\nFirst Name: %s\n", emp1.fname);
printf("Last Name: %s\n", emp1.lname);
printf("Employee ID: %d\n", emp1.id);
printf("Salary: %.2f\n", emp1.salary);
}
```

Advanced data types/Data structures

The keywoed *typedef*

The keywoed *typedef* is used to create structure definitions to build an alias relationship with the structure tag (structure name). It gives a shortcut when creating instances of the structure:

```
typedef struct employee { //(modified)
char fname[10];
char lname[10];
int id;
float salary;
} emp; //(modified)
main( )
{
//create instance of employee structure using emp
emp emp1; //(modified)
```

Advanced data types/Data structures

```
//assign values to members
strcpy(emp1.fname, "Bibhas");
strcpy(emp1.lname, "Adhikari");
emp1.id = 123;
emp1.salary = 50000.00;
//print member contents
printf("\nFirst Name: %s\n", emp1.fname);
printf("Last Name: %s\n", emp1.lname);
printf("Employee ID: %d\n", emp1.id);
printf("Salary: %.2f\n", emp1.salary); }
```

Advanced data types/Data structures

Arrays of Structures

Declaring and working with array of structure is same as declaring and working an array of fundamental data types, such as integer, character, or float

```
typedef struct employee {
char fname[10];
char lname[10];
int id;
float salary;
} emp;
emp emp1[5]; //(modified)
strcpy(emp1[0].fname, "Bibhas");
```

Advanced data types/Data structures

Example of Arrays of Structures:

```
typedef struct scores {
char name[10];
int score;
} s;
main( )
{
s highScores[3];
int x;
//assign values to members
strcpy(highScores[0].name, "Ankush");
highScores[0].score = 40768;
strcpy(highScores[1].name, "Bibhas");
highScores[1].score = 38565;
strcpy(highScores[2].name, "Chandan");
highScores[2].score = 35985;
```

Advanced data types/Data structures

Example of Arrays of Structures (contd.):

```
//print array content
printf("\nTop 3 High Scores \n");
for (x = 0; x < 3; x++)
printf("\n%s\t%d\n", highScores[x].name,
highScores[x].score);
}
```

Advanced data types/Data structures

Passing structures to functions

- Passing structure by value to a function: the function prototype and function definition are to be supplied with the structure tag
- Passing structures by reference: members of structures can be accessed via the structure pointer operator $(->)$, no gap between $-$ and $>$

ADT:Passing structure by value to a function

```
typedef struct employee {
int id;
char name[10];
float salary;
} e;
void processEmp(e); //supply prototype with structure alias name
main( )
{
e emp1 = {0, 0, 0}; //Initialize members
processEmp(emp1); //pass structure by value
printf("\nID: %d\n", emp1.id);
printf("Name: %s\n", emp1.name);
printf("Salary: $%.2f\n", emp1.salary);
}
```

ADT:Passing structure by value to a function

```
void processEmp(e emp) //receives a copy of the structure
{
emp.id = 123;
strcpy(emp.name, "Sheila");
emp.salary = 65000.00;
} //end processEmp
```

ADT:Passing structure by reference

Structure pointer operator:

```
main( )
{
typedef struct player {
char name[15];
float score;
} p;
p aPlayer = {0, 0}; // create instance of structure
p *ptrPlayer; // create a pointer of structure type
ptrPlayer = &aPlayer; // assign address to pointer of structure type
strcpy(ptrPlayer− >name, "Bibhas Adhikari"); // access
through indirection
ptrPlayer− >score = 1000000.00;
printf("\nPlayer: %s\n", ptrPlayer− >name);
printf("Score: %.0f\n", ptrPlayer− >score);
}
```

ADT:Passing structure by reference

Using structure pointer operator inside function for accessing members of the structure:

```
typedef struct employee {
int id;
char name[10];
float salary;
} emp;
main( )
{
emp emp1 = {0, 0, 0};
emp *ptrEmp;
ptrEmp = &emp1;
processEmp(ptrEmp);
printf("\nID: %d\n", ptrEmp->id);
printf("Name: %s\n", ptrEmp->name);
printf("Salary: $%.2f\n", ptrEmp->salary);
}
```

ADT:Passing structure by reference

```
void processEmp(emp *e)
{
e->id = 123;
strcpy(e->name, "Bibhas");
e->salary = 65000.00;
}
```