CS19001: Programming and Data Structures Laboratory

Soumyajit Dey, Aritra Hazra
CSE, IIT Kharagpur

01-Oct-2018

Multi-dimensional Arrays

One-dimensional arrays are quite able to represent many natural collections. There are some other natural collections that may better be conceptualized as 2-dimensional data. Example: a matrix.

$$\begin{matrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{matrix}$$

Two-Dimensional Arrays

```
int a[3][4] = {
 {0, 1, 2, 3} ,    /*for row index 0 */
 {4, 5, 6, 7} ,    /*for row index 1 */
 {8, 9, 10, 11}    /*for row index 2 */
};
```

or

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

Matrix addition

```
int m, n, c, d;
int first[9][9], second[9][9], sum[9][9];
printf("Enter no. of rows and columns\n");
scanf("%d%d", &m, &n);
printf("Enter the elements of matrix1\n");
for ( c = 0 ; c < m ; c++ )
  for ( d = 0 ; d < n ; d++ )
    scanf("%d", &first[c][d]);
/*now scan 2nd matrix*/

for ( c = 0 ; c < m ; c++ )
  for ( d = 0 ; d < n ; d++ )
    sum[c][d] = first[c][d] + second[c][d];
```

Structures

Structure is a user defined data type available in C, which allows you to combine data items of different kinds.

```
struct Books
{
   char   title[50];
   char   author[50];
   char   subject[100];
   int    book_id;
};
```

```
int main( )
{
   struct Books Book1;
   /* Declare Book1 of type Book */
   struct Books Book2;
   /* Declare Book2 of type Book */

   /* book 1 specification */
   strcpy( Book1.title, "C Programming");
   strcpy( Book1.author, "Nuha Ali");
   strcpy( Book1.subject, "C Programming");
   Book1.book_id = 6495407;

   /* book 2 specification */
   strcpy( Book2.title, "Telecom Billing");
   strcpy( Book2.author, "Foolan Barik");
   strcpy( Book2.subject, "Telecoms");
   Book2.book_id = 6495700;
/* print Book1 info */
   printf("Bk1 title %s\n", Book1.title);
   printf("Bk1 author %s\n", Book1.author);
   printf("Bk1 sub %s\n", Book1.subject);
   printf("Bk1 id %d\n", Book1.book_id);

   /* print Book2 info */
   printf("Bk2 title %s\n", Book2.title);
   printf("Bk2 author %s\n", Book2.author);
   printf("Bk2 sub %s\n", Book2.subject);
   printf("Bk2 id %d\n", Book2.book_id);

   return 0;
}
```

Structures as Function Arguments

```
struct Books
{
   char  title[50];
   char  author[50];
   char  subject[100];
   int   book_id;
};/* function declaration */
void printBook( struct Books book );
int main( )
{ //after book specification
  /* print Book1 info */
  printBook( Book1 );
  /* Print Book2 info */
  printBook( Book2 );
  return 0;
}
```

Structures as Function Arguments

```
void printBook( struct Books book )
{
   printf("Book title: %s\n", book.title);
   printf("Book author: %s\n", book.author);
   printf("Book sub: %s\n", book.subject);
   printf("Book id: %d\n", book.book_id);
}
```

- The function body

Array of Structures

```c
struct inventory {
      int part_no;
      float cost;
      float price;
};

struct inventory table[4];
```

Programming Assignments

Complete and submit during lab

Assignment 1 [PDS-Restaurant]

Write a main program to define an array of 50 structures named "order" to handle order processing in a PDS-restaurant. The structure has the following members.

- `char name[11]` : string of 10 character to store customer name.
- `int item[5]` :quantity of an item ordered by the customer (there are five items).

There also exists another array `itemRevenue[5]` storing 'revenue (profit) per item' per plate of all five available items. This is also a user input. Apart from that user also inputs another value (x), which represents 'minimum target average revenue (profit) for consecutive three orders' (the meaning and usage of this is explained in details later!).
In an infinite loop, look for any of the following user specified character choices/commands (a, b, c, d, e) and perform respective tasks as listed below. Exit if the user enters 'e'.

Assignment 1 [PDS-Restaurant]

- (Choice: a) `create-new-order`: This case should ask the name of the customer and the quantity of each of the five items. Insert the order at the end of the pending orders and print all the pending orders (customer name and items ordered and corresponding quantities) till now. Each order is to be given a sequentially increasing unique ID (denote by the index of the array of structures).
- (Choice: b) `delete-existing-order`: This case requests the user to specify an order by its ID or by the name of the customer (first ask to input for ID and if forgotten (-ve number is entered) then customer-name is asked). Here, you should display the existing order to be deleted and ask for confirmation. Upon confirmation (Y/N), delete the order and move the remaining orders forward in the structure-array.

Assignment 1 [PDS-Restaurant]

- (Choice: c) `serve-order`: In this case, you should display and delete the order that was placed the earliest and move the remaining orders forward in the structure-array. Immediately, it should also display all pending orders till now.
- (Choice: d) `display-item-statistics`: Display which item in what quantities are required to meet all the pending requests.
- (Choice: e) `terminate`: Exit from the infinite loop and terminate execution of the program.

Assignment 1 [PDS-Restaurant]

Justification of the value in x indicating 'minimum target average revenue (profit) for consecutive three orders':

- Consider the fact that PDS-restaurant owner may be greedy. He has a revenue (profit) model stating 'every three *consecutive* orders' in the pending orders gives an average revenue of at least x.
- Take x as user input.
- Implement this added `check-order` part inside `create-new-order` case (i.e. under Choice: a) which assumes that the owner's requirement holds for currently pending orders.
- A new order is accepted only if there may be a valid position in which a new-coming order can be inserted so that the revenue (profit) model is satisfied. Otherwise, the new order is rejected (not taken).

Assignment 1 [PDS-Restaurant]

- The *greedy owner* can insert a new order anywhere in the list in order to satisfy the revenue model !!!
- Example: Let the current pending orders be generating revenues of $(10.0, 8.25, 9.75, 5.0)$. Note that if $x = 7.5$, we have every three consecutive order giving an average revenue > 7.5 which is fine. Let a new order is paced where the revenue is computed as 5.5 for all the items ordered. Note $(10.0, 8.25, 9.75, 5.0, 5.5)$ do not satisfy the revenue model but $(10.0, 5.5, 8.25, 9.75, 5.0)$ does. Hence, the function will enter the order as the second element in the pending list!

NOTE: This `check-order` part may be implemented once user places new order from choice-a. It is recommended that, you first complete the assignment without this checking, and then add this part (separately)!

Thank You