MA 511: Computer Programming

Lecture 7: Recursion, Structure, Union

http://www.iitg.ernet.in/psm/indexing_ma511/y10/index.html

Partha Sarathi Mandal

psm@iitg.ernet.ac.in

Dept. of Mathematics, IIT Guwahati

Semester 1, 2010-11

Recursion

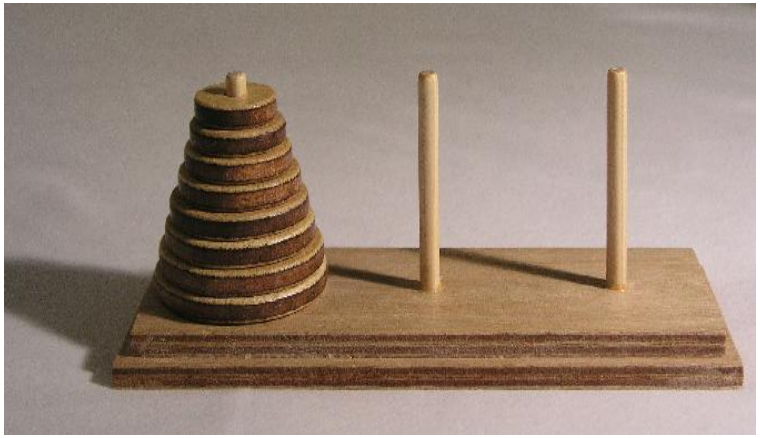
- Is a process by which a function calls itself repeatedly until some specific condition has been satisfied.

Example:

```
long int factorial(int n){
        If(n<=1)
            return(1);
        else
            return(n*factorial(n-1));
}
main(){
    int n;
    print("Type n = ");
    scanf("%d", &n);
    printf("n! = %ld ", factorial(n));
}
```

The Towers of Hanoi

- It consists of three rods, and a number of disks of different sizes which can slide onto any rod.
- The puzzle starts with the disks neatly stacked in order of size on one rod, the smallest at the top, thus making a conical shape.
- The objective of the puzzle is to move the entire stack to another rod, obeying the following rules:
  - Only one disk may be moved at a time.
  - Each move consists of taking the upper disk from one of the pegs and sliding it onto another rod, on top of the other disks that may already be present on that rod.
  - No disk may be placed on top of a smaller disk.



The Towers of Hanoi

- Can you write a c program for the **Towers of Hanoi** using *recursion* where number of disks is input ?

Assignments

1. Write a C-program for the function f(x) defined below:

$$f(x) = 2x^2 + 3x + 4 \quad \text{for } x < 2$$
$$= 0 \quad \text{for } x = 2$$
$$= -2x^2 + 3x - 4 \quad \text{for } x > 2$$

2. Write function to validate the elements of a matrix of size nxn. The validation rules are:
   a) All diagonal entries should be positive.
   b) The matrix should be symmetric.
   c) All the non-diagonal elements should be negative or zero.

structures

array: Data Structure whose elements are all of the same data type.

Example:

int A[10]; float B[10];

structure: individual elements can differ in type:

Example:

```
struct account {
        int    acct_no;
        char  acct_type;
        char  name[80];
        float  balance;
};
```

where acct_no, acct_type, name[80], balance are the member of the structure **account.**

Contd.. structures

Example:

```
struct account {
    int acct_no;
    char acct_type;
    char name[80];
    float balance;
} oldcustomer, newcustomer;
```

- oldcustomer, newcustomer are the structure variable of type account.

Contd.. structures

Example:

```
struct account {
        int acct_no;
        char acct_type;
        char name[80];
        float balance;
};
```

- We also can declare the structure variable follows:
- struct account oldcustomer, newcustomer;
- oldcustomer, newcustomerare the structure variable of type account.

Contd.. structures

Example:

```
struct account {
    int acct_no;
    char acct_type;
    char name[80];
    float balance;
}customer[100];
```

- This declarations implies that **customer** is a **100** element array of the structures of the type account.

Example

how to access variables acct_no, acct_type, name, balance of the **struct** account ?

```
struct account {
        int acct_no;
        char acct_type;
        char name[80];
        float balance;
};
```

```
static/external  struct account customer[ ] = {12, 'S', "abcd", 123.0,
                                               13, 'C', "wert", 234.0,
                                               14, 'S', "rsdef", 1234.0};
```

or

```
static/external  struct account customer[ ] = {{12, 'S', "abcd", 123.0}, {13, 'C', "wert",
    234.0},{14, 'S', "rsdef", 1234.0}};
```

static/external : storage class

Example

how to access variables acct_no, acct_type, name, balance of the **struct** account ?

```
#include<stdio.h>
void input_data(int i);
void output_data(int i);
struct account {
        int acct_no;
        char acct_type;
        char name[80];
        float balance;
} customer[100];
main(){     int i, n;
        printf("No of Customers? ");
        scanf("%d", &n);
        for(i = 0; i < n; i = i + 1)
            input_data(i);
        for(i = 0; i < n; i = i + 1)
            output_data(i);
} //end of the main
```

```
void input_data(int i){
        printf("Name of the customer:");
        scanf(" %[^\n]", customer[i].name);
        printf("Acct_no of the customer :");
        scanf("%d", &customer[i].acct_no);
        printf("Balance of the customer :");
        scanf("%f", &customer[i].balance);
        printf("Type of the customer ");
        scanf(" %c", &customer[i].acct_type);
}

void output_data(int i){
    printf("\n Name of the customer:  %s",  customer[i].name);
    printf("\n Acct_no of the customer :  %d", customer[i].acct_no);
    printf("\n Balance of the customer :  %f", customer[i].balance);
    printf("\n Acct_type of the customer : %c", customer[i].acct_type);
}
```

Exercise

- Write a c-programming using **struct** for detecting a set of given *n* randomly generated points are belonging to a given circle or not.

```
struct coordinates {
        int x;
        int y;
} points[100];
```

typedef

int i, j; equivalent to **typedef** int mydef;
mydef i, j;

```
typedef struct {
        int acct_no;
        char acct_type;
        char name[80];
        float balance;
} account ;
account oldcustomer, newcustomer;
```

Member of a struct may be a struct

```
typedef struct {
        int day;
        int month;
        int year;
        } date;
typedef struct {
        int acct_no;
        char acct_type;
        char name[80];
        float balance;
        date update;
} customer[100] ;
```

Equivalent to:

```
struct date {
                int day;
                int month;
                int year;
                };
struct  account {
                int acct_no;
                char acct_type;
                char name[80];
                float balance;
                struct date update;
};
struct  account  customer[100];
```

customer[i].acct_no: variable of the structure account

customer[i].update.month: variable of the **structure** **date**

Union

```
Union tag {
    member 1;
        …
    member m;
};
```

```
Union account {
        int acct_no;
        char acct_type;
        char name[80];
        float balance;
};
```

- Like structures, contain members whose individual data types may differ from one another.
- union allocates the memory equal to the maximum memory required by the member of the union but structure allocates the memory equal to the total memory required by the members.
- In union, one block is used by all the member of the union but in case of structure, each member have their own memory space
- Union is useful for application where values need not be assigned to all of the members simultaneously.