Lecture05: Arrays & Recursion
3/25/2013

Slides modified from Yin Lou, Cornell CS2022: Introduction to C

1

Review

- Basic Types: {`short, int, long, long long, float, double, long double, char`}
- Function:

```
double area(double radius)
{
            return(3.1415 * radius * radius);
};
```

- Loop: {`for (i=1; i<=5; i++) …, while … }`
- Condition: {`if, else if, switch, case, … }`

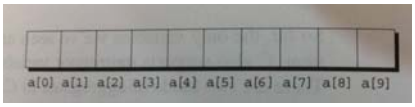
- New: Arrays & Recursion

2

Arrays

- Each variable can store only one value. What if you want a variable to store **many** values?
- To declare an array, use []:

```
int a[10]; // create an array with 10 integer elements
double b[5]; // create an array with 5 double elements
```



- Common to use a constant (MACRO) to define the length of an array

```
#define N 10
int a[N];
```

3

More on Arrays

- For most compilers, the size of an array **cannot** be changed after declaration
  - The number between the brackets must be a constant
- You can give initial values for array elements:

```
int a[5] = {3, 7, -1, 4, 6};
```

- A better way:

```
int a[] = {3, 7, -1, 4, 6}; // Let the compiler
calculate the size
```

- Sizeof()

```
char a[] = {'a', 'b', 'c'};
for (i=0; i<sizeof(a); i++)
    a[i] = 0;
```

4

Access Array elements

- Array indices in C are zero-based, e.g. a[0], a[1], ..., a[4]

```
void main()
{
       int a[] = {3, 7, -1, 4, 6};
       int i;
       double average = 0;

       // compute the average of values in an array
       for (i = 0; i < 5; ++i)
       {
              average += a[i];
       }
       average /= 5;  // average = average / 5;
       printf("Average = %.2f\n", average);
}
```

5

More Array Examples

- What do the following code do?

```
for (i = 0; i < N; i++)
       a[i] = 0;

for (i = 0; i < N; i++)
       scanf("%d", &a[i]);

for (i = 0; i < N; i++)
       sum += a[i];

____

int a[10], i;
for (i = 0; i <= 10; i++)
       a[i] = 0;
```

6

In-Class Exercise 5-1

Write a program that asks the user to enter 5 integers (store these 5 integers in an array), then writes the numbers in reverse order:

```
Enter 5 integers: 34 82 49 102 7
In reverse order: 7 102 49 82 34
```

7

Multidimensional Arrays

```
/* m[0][0] = 1; m[0][1] = 1; m[0][2] = 1;
 * m[1][0] = 0; m[1][1] = 0; m[1][2] = 0;
 */
int m[2][3] =     {1, 1, 1, 0, 0, 0};

int m[2][3] =     {{1, 1, 1}, {0, 0, 0}};


m[2][1] = 0;
```

8

Recursion – factorial number

- A function is recursive if **it calls itself**.

```
int fact(int n)
{
   if (n <= 0)
   {
        return 1;
   }
   else
   {
        return n * fact(n - 1);
   }
}

fact(3);
```

$$\text{fact}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot \text{fact}(n-1) & \text{if } n > 0 \end{cases}$$

fact(3) calls fact(2)
  fact(2) calls fact(1)
    fact(1) returns 1 to fact(2)
  fact(2) returns 2*1 = 2 to fact(3)
fact(3) returns 3*2=6

9

Recursion – Fibonacci number

```
int fib(int n)
{
   if (n == 0)
        return 0;
   else if (n == 1)
        return 1;
   else
        return fib(n-2) +
               fib(n-1);
}

fib(3);
```

$$fib(n) = \begin{cases} 0 \; if \; n = 0 \\ 1 \; if \; n = 1 \\ fib(n-1) + fib(n-2) if \; n > 1 \end{cases}$$

fib(3) calls fib(1) + fib(2)
  fib(1) returns 1
  fib(2) calls fib(0) + fib(1)
    fib(0) returns 0
    fib(1) returns 1
  fib(2) returns 0+1=1
fib(3) returns 1+1=2

10

In-Class Exercise 5-2

Can you think about other recursive examples?

Write a **recursive** function to calculate the sum of integers $\sum_{i=1}^{n} i$ starting from 1 to n.

```
Enter an integer> 5
15

Enter an integer> 2
3
```

11

Why Recursion?

- It comes naturally with a common CS problem solving approach called **Divide-and-Conquer**.
  - What is divide-and-conquer?
  - Divide a large (complex) problem into small (simple) subproblems ; then solve the subproblems.
- Classical example: sorting

12

Sorting algorithm

- Enter 10 numbers into an array. Sort the array in an increasing order and print the array.

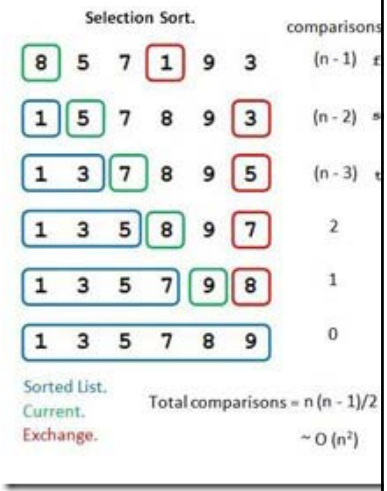
```
Enter 6 numbers to be sorted: 8 5 7 1 9 3
In sorted order: 1 3 5 7 8 9
```

- How to solve this problem?
- How to solve this problem using recursion?

13

Selection Sort – how to solve this problem?

- Start from the 1st array element
- Find the smallest number in the array[0..n]
- Swap the smallest number with the 1st array element
- Continue with the 2nd array element
- Find the smallest number in array[1..n]
- Swap the smallest number with 2nd array element
- Continue till the last array element
- **How to solve this problem using recursion?**



14

Selection Sort – how to solve this recursively?

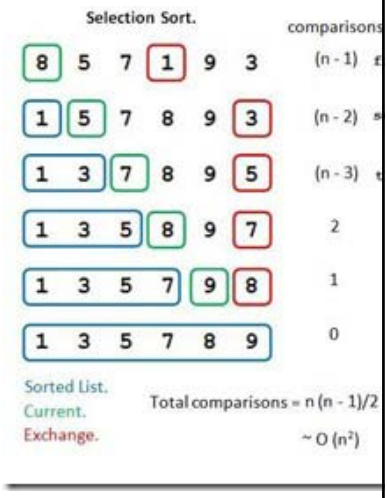
n is the size of unsorted list.

if n is 1,

- array is sorted

else

- (1) find the smallest element in the unsorted array list
- (2) swap the 1st element in the unsorted array with the smallest element
- (3) sort the remaining unsorted array



15

In-Class Exercise 5-3(a-d)

Code Selection Sort algorithm described in the previous slide.

```
Enter 6 numbers to be sorted : 8 5 7 1 9 3
1 3 5 7 8 9
```

Define the following recursive functions:

```
/* 5-3(a): print the array from a[i] to a[n] */
int print_array(int a[], int i, int n);

/* 5-3(b): swap array elements a[i] <-> a[j] */
void swap_element(int a[], int i, int j);

/* 5-3(c): find the smallest element in a[i..n] and returns its
array index */
int find_sm_ind(int a[], int i, int n);

/* 5-3(d): Sort the unsorted list a[i..n] by using prev
 * functions: (1) find_sm_ind, (2) swap, (3) recurse */
void selection_sort(int a[], int i, int n);
```

16

In-Class Exercise 5-3: Skeleton code

```
void print_array(int a[], int i, int n);
void swap_element(int a[], int i, int j);
int find_sm_ind(int a[], int i, int n);
void selection_sort(int a[], int i, int n);

int main()
{
    int a[6], i;
    printf("Input 6 numbers to be sorted : ");
    for (i=0;i<6;i++)
        scanf("%d", &(a[i]));

    // swap_element(a, 2, 4);
    // printf(“test find_sm_ind %d\n”, find_sm_ind(a, 2, 5));
    // selection_sort(a, 0, 5);
    print_array(a, 0, 5);
    return 1;
}
```

17