

COMP20005 Workshop Week 7

Preparation:

open `grok`, `jEdit`, and `minGW` (or `Terminal` if yours is a `Mac`)

- | | |
|---|---|
| 1 | Arrays: concept, representation, usage |
| 2 | Discussion 1: Exercise 3.06 revisited → <i>an advantage of using arrays</i> |
| 3 | Discussion 2: Ex 7.4 → <i>Conventional ways to work with arrays</i> → <i>Redirection:</i> reading data from a file (instead of from the keyboard) → <i>Selection Sort</i> (don't confuse with <i>Insertion Sort</i>) |
| 4 | A Case Study based on exercise 7.5 → Using <code>struct</code> and <code>typedef</code> |
| 5 | Assignment 1: Q&A |
| 6 | Lab: <ul style="list-style-type: none">• do your assignment 1 and test the submission, or• do exercises 7.1, 7.7 – 7.10 |

Arrays

Situation:

We need to manipulate a series of number like 1, 4, 10, 3, 4, 6
Mathematically speaking, we are working with the sequence

$x_1, x_2, x_3, x_4, x_5, x_6$

where

$$x_1 = 1$$

$$x_2 = 4$$

...

$$x_6 = 4$$

With $i=3$ we can say that x_i has the value of 10. We even can:

set S to 0

for i from 1 to 6

add x_i to S

Quite convenient!

Can we do a similar thing in C?

Arrays

Situation:

We need to manipulate a series of number like 1, 4, 10, 3, 4, 6
Mathematically speaking, we are working with the sequence

$x_1, x_2, x_3, x_4, x_5, x_6$

Can we do a similar thing in C?

YES. Here, we will declare `x` as an *array* of 6 `int` elements.

```
int x[6];
```

then:

the first element of `x` is referred to as `x[0]`

the second element of `x` is referred to as `x[1]`

the `i`-th element of `x` is referred to as `x[i]`, with `0 ≤ i < 6`






and we can do the assignments:

```
x[0] = 1;
```















```
x[1] = 4;
```

```
...
```






















arrays: declaration & use

| | statements | variables in memory (<i>after</i> LHS statements) |
|---|---|---|
| 1 | <code>int i, A[5];</code> /* equivalent to declaring 5 variables, each is of data type <code>int</code> */ | <div> <div>i</div> <div>A[0]</div> <div>A[1]</div> <div>A[2]</div> <div>A[3]</div> <div>A[4]</div> </div>  |
| 2 | <code>A[0] = 10;</code> <code>i = A[0] * 2;</code> |  |
| 3 | <code>i = 2;</code> <code>A[i] = 20;</code> |  |
| 4 | <code>for (i=0; i<5; i++) {</code> <code>A[i] = i*i;</code> <code>}</code> |  |
| 5 | <code>for (i=0; i<3; i++) {</code> <code>scanf ("%d", &A[i]);</code> <code>}</code> /* supposing that input from keyboard is 10 20 30 */ |  |

arrays...

| | statements | variables in memory (after LHS statements) | | | | | | |
|---|--|--|---|---|---|---|---|---|
| 1 | <code>int i, sum=0, A[5]= {0,1,2,3,4};</code> | <code>i</code> | <code>sum</code> | <code>A[0]</code> | <code>A[1]</code> | <code>A[2]</code> | <code>A[3]</code> | <code>A[4]</code> |
| | |  |  |  |  |  |  |  |
| 2 | <code>for (i=0; i<5; i++) sum += A[i];</code> |  |  |  |  |  |  |  |

arrays...

| | statements | variables in memory (after LHS statements) | | | | | | |
|---|---|--|---|---|---|---|---|---|
| 1 | <code>int i, sum=0, A[5]= {0,1,2,3,4};</code> | i | sum | A[0] | A[1] | A[2] | A[3] | A[4] |
| | |  |  |  |  |  |  |  |
| 2 | <code>for (i=0; i<5; i++) sum += A[i];</code> |  |  |  |  |  |  |  |
| 3 | <code>for (i=0; i<4; i++) { A[i+1]= A[i]; }</code> |  |  |  |  |  |  |  |

arrays...

| | statements | variables in memory (after LHS statements) | | | | | | |
|---|---|---|-----|------|------|------|------|------|
| 1 | <code>int i, sum=0, A[5]= {0,1,2,3,4};</code> | i | sum | A[0] | A[1] | A[2] | A[3] | A[4] |
| | | 0 | 0 | 1 | 2 | 3 | 4 | |
| 2 | <code>for (i=0; i<5; i++) sum += A[i];</code> | 5 | 10 | 0 | 1 | 2 | 3 | 4 |
| 3 | <code>for (i=0; i<4; i++) { A[i+1]= A[i]; }</code> | 0 | 10 | 0 | 0 | 0 | 0 | 0 |

Notes: No operation with *whole arrays* is allowed. With declaration:

```
int A[3]={10,20,30}, B[3];
```

we cannot write:

```
B= A;
```

```
if (A==B) A= A+B;
```

Arrays : using in C



```
#define SIZE 5
...

int X[SIZE]= {1, 2, 3, 4, 5};
int n= SIZE;
```

In computer memory, an array is stored as *a block of contiguous cells*, one cell for one array's element.

Essentially, an array is defined by 4 objects:

- **X**: the array's name, which is actually a pointer to the memory block
- **int**: the data type of each element of the array
- **SIZE**: an **int** constant representing the array's size
- **n**: a buddy **int** variable, representing the number of elements that are currently employed

Arrays as function arguments

- With a function prototype, say:

```
int change_array(int A[], int n);
```

we should note that:

- the formal parameter `A[]` is an array of `int`, but no size is specified in “`int A[]`”,
 - instead, there is another parameter, `n`, which specifies the *current* size of `A[]`,
 - the array formal parameter `A[]` is an array, is a pointer, so it can imply *both input and output* of function `change_array`.
- With the call “`change_array(B, 10)`”:
 - the action in formal parameter `A[]` is actually happen to `B[]`.
 - What do the following call mean:

```
int X[10]= {1,2,4,2,3,3,1,1,9,8};
```

```
int n= 10;
```

```
change_array(B, 10);
```

```
change_array(&B[0], 10);
```

```
change_array(B, 12);
```

```
change_array(&B[6], 4);
```

```
change_array(B+6, 4);
```

Discussion 1: Exercise 3.06 revisited

Discussion 2: A Case Study, based on exercise 7.5

Suppose that a set of "student number, mark" pairs are provided, one pair of numbers per line, with the lines in no particular order. Write a program that reads this data and outputs the same data, but ordered by student number. For example:

823678 66

765876 94

864876 48

785671 68

854565 89

On this input your program should output:

Enter as many as 1000 "studnum mark" pairs, ^D to end

5 pairs read into arrays

studnum mark

765876 94

785671 68

823678 66

854565 89

864876 48

Hint: use two parallel arrays, one for student numbers, and one for the corresponding marks. You may assume that there are at most 1,000 pairs to be handled.

testing

Assignment 1: Q&A

Assignments: advices

- *Be active in the subject's Discussion Forum!*
- *Visit **LMS→Assignment 1** frequently!*
- *Read the specifications carefully.*
- *Read the marking rubric carefully and try to maximize your marks!*
- *Read the **sample solution to 2015** (in LMS.Assignment1, point 6), focusing on **main()**. You can learn something from there.*
- *Check your program carefully, at least with all supplied data files. Make sure that your outputs are the same as the expected outputs.*
- *Make as many submissions as you want, only the last one (before deadline) counts. Do verify and read the report from verify to make sure that your program works well on dimefox.*
- ***START EARLY, START RIGHT NOW! SUBMIT EARLY, SUBMIT EVERY DAY!***