Computer Programming 1 Lab

2020-10-29

Chang, Chi-Hung

Outline

- OJ-CLI
- Array
  - Sort Array
  - Search in Array
- Pointer
- const
- Exercise 6

OJ-CLI

OJ-CLI

Installation

- Ghost Server

```
echo 'export PATH="~f103207425/.local/bin:$PATH"' >> ~/.profile
source ~/.bashrc
```

- Install from Source Code
  i. Clone this project

```
git clone https://github.com/josix/oj-cli.git
```

  ii. Update `HOST` variable value in file `oj-cli/constants.py` to the OnlineJudge URL you accessing

  iii. Update Shebang(`#!/opt/csw/bin/python2.7`) value to a suitable one in `oj.py`

OJ-CLI

Commands

- `oj login`
  - Use `oj login` to login to the account in OnlineJudge. It required you to enter your account information so that `oj-cli` could access OnlineJudge service successfully. After entering your username and password. `oj-cli` will respond if you login successfully or not.

```
$ oj login
Username:
Password:
```

OJ-CLI

Commands

- `oj get_assign <assign_no>`
  - Use `oj get_assign <assign_no>` to download the latest assignment from contest. The downloaded files are stored in folder `hwX` or `exX`. The folder includes testing data, output data, and template C script, which are named as `1.in`, `1.out`, and `hwX.c` (or `exX.c`) separately.

```
$ oj get_assign hw2
$ oj get_assign ex2
```

OJ-CLI

Commands

- `oj submit <assign_no> <code_file>`
  - Use oj `submit <assign_no> <code_file>` to submit your code to contest.

```
$ oj submit hw2 hw2.c
$ oj submit ex3 ../ex3.c
```

Array

- Sort Array
- Search in Aray

Array

- Sort Array

  Bubble sort

```
int SIZE = 5;
int array[SIZE] = {2, 3, 5, 1, 4};
for(int i = 0 ; i < SIZE - 1 ; i++){
    for(int j = 0 ; j < SIZE - i - 1 ; j++){
        if(array[j] > array[j+1]){
            int temp = array[j];
            array[j] = array[j+1];
            array[j+1] = temp;
        }
    }
}
```

Array

- Search in Array
  - **Linear search**

```
int SIZE = 5;
int array[SIZE] = {2, 3, 5, 1, 4};
int target = 4;
for(int i = 0 ; i < SIZE ; i++){
    if(array[i] == target){
        printf("index = %d\n", i);
        break;
    }
}
```

Array

- Search in Array
  - **Binary search**

```
int binarySearch(const int array[], int target, int low, int high){
    int middle;
    while(low <= high){
        middle = (low + high) / 2;
        if(target == array[middle]){
            return middle;
        }
        else if(target < array[middle]){
            high = middle - 1;
        }
        else{
            low = middle + 1;
        }
    }
    return -1;
}
```

- Search in Array
  - **Binary search**

```c
int SIZE = 8;
int array[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8};
int target = 7;
printf("%d\n", binarySearch(array, target, 0, SIZE-1);  // 6
```

STEP	---1---	---2---	---3---	---4---	---5---	---6---	---7---	---8---
(1)	L							H
				(M)				
(2)					L			H
						(M)		
(3)							L	H
							(M)	

Pointer

Pointer

```
void main(){
    int a = 1;
    int b = 2;
    int c = 3;
}
```

Memory Address	Value	Variable
0X0012FF70	1	a
0X0012FF74	2	b
0X0012FF78	3	c

Pointer

- `*`
  i. Pointer (□□)
     Declare that the type of the variable is a pointer. → It stores **address**.
  ii. Dereference operator (□□□□□)
     Apart from variable declaration, we use `*` to get the value which is stored in the variable's address.
- `&`
  Address-of operator (□□□□□) → Get the variable's **memory address**.

Pointer

```
void main(){
    int a = 1;
    int* ptr = &a;  // Declare a int pointer named "ptr" and it points to a's address
```

Memory Address	Value	Variable
0X0012FF70	1	a
0X0012FF74	0X0012FF70	ptr

- **Pointer variable**: a variable that stores pointer
- **Pointer**: point to a variable's address

const

const

- `int* ptr;`
  - □□□□□□□□□□□
  - □□□□□□□□□□□
- `int* const ptr;`
  - □□□□□□□□□□□
  - □□□□□□□□□□□
- `const int* ptr;`
  - □□□□□□□□□□□
  - □□□□□□□□□□□
- `const int* const ptr;`
  - □□□□□□□□□□□
  - □□□□□□□□□□□

Exercise 6

Matrix Multiplication

$$\begin{bmatrix} 5 & 8 & -4 \\ 6 & 9 & -5 \\ 4 & 7 & -2 \end{bmatrix} \times \begin{bmatrix} 2 \\ -3 \\ 1 \end{bmatrix} = \begin{bmatrix} -18 \\ -20 \\ -15 \end{bmatrix}$$

- Input: Pairs of matrices we want to multiple until end-of-file. Each pair has two matrices.
- Output: Print the multiplied matrices, or print "Invalid calculation!!" if two matrices cannot be multiplied.

Get repository on Ghost: `oj get_assign ex6`
Submit on Ghost: `oj submit ex6 <code_file>`
(Remember to login first!!)

Any Questions?