

# Program Assignment 1

2021/03/18 by Mark Chang

# Outlines

- CPP Tutorial
- PA1 Introduction

# CPP Tutorial

- <https://www.tutorialspoint.com/cplusplus/index.htm>
- CPP Variable
- CPP Pointer
- CPP Dynamic Memory
- CPP Class & Object
- Pointer to Classes

# CPP Variable

- [https://www.tutorialspoint.com/cplusplus/cpp\\_variable\\_types.htm](https://www.tutorialspoint.com/cplusplus/cpp_variable_types.htm)
- Variable definition:
  - type variable\_name;
  - type variable\_name = value;
- Variable initialization:
  - variable\_name = value;

```
// Variable definition:  
int a, b;  
int c;  
float f;  
  
// actual initialization  
a = 10;  
b = 20;  
c = a + b;  
  
cout << c << endl ;  
  
f = 70.0/3.0;  
cout << f << endl ;
```

```
30  
23.3333
```



# CPP Pointer

- [https://www.tutorialspoint.com/cplusplus/cpp\\_pointers.htm](https://www.tutorialspoint.com/cplusplus/cpp_pointers.htm)
- Variable address:
  - & variable\_name

```
int var1;  
char var2[10];  
  
cout << "Address of var1 variable: ";  
cout << &var1 << endl;  
  
cout << "Address of var2 variable: ";  
cout << &var2 << endl;
```

```
Address of var1 variable: 0xbfefd5c0  
Address of var2 variable: 0xbfefd5b6
```

# CPP Pointer

- [https://www.tutorialspoint.com/cppplusplus/cpp\\_pointers.htm](https://www.tutorialspoint.com/cppplusplus/cpp_pointers.htm)
- Pointer
  - `type* pointer_name;`
  - `type pointer_name = &variable_name;`
- Pointer value
  - `*pointer_name`

```
int var = 20;    // actual variable declaration.
int *ip;         // pointer variable

ip = &var;       // store address of var in pointer variable

cout << "Value of var variable: ";
cout << var << endl;

// print the address stored in ip pointer variable
cout << "Address stored in ip variable: ";
cout << ip << endl;

// access the value at the address available in pointer
cout << "Value of *ip variable: ";
cout << *ip << endl;
```

```
Value of var variable: 20
Address stored in ip variable: 0xbfc601ac
Value of *ip variable: 20
```

# CPP Dynamic Memory

- [https://www.tutorialspoint.com/cplusplus/cpp\\_dynamic\\_memory.htm](https://www.tutorialspoint.com/cplusplus/cpp_dynamic_memory.htm)
- Allocate new memory
  - `pointer_name = new type;`
- Free up the memory
  - `delete pointer_name;`

```
double* pvalue = NULL; // Pointer initialized with null
pvalue = new double;    // Request memory for the variable

*pvalue = 29494.99;      // Store value at allocated address
cout << "Value of pvalue : " << *pvalue << endl;

delete pvalue;           // free up the memory.
```

```
Value of pvalue : 29495
```

# CPP Class & Object

- [https://www.tutorialspoint.com/cppplus/cpp\\_classes\\_objects.htm](https://www.tutorialspoint.com/cppplus/cpp_classes_objects.htm)

## Class

```
class Box {  
    public:  
        double length;    // Length of a box  
        double breadth;   // Breadth of a box  
        double height;    // Height of a box  
};
```

## Object

```
Box Box1;        // Declare Box1 of type Box  
Box Box2;        // Declare Box2 of type Box  
double volume = 0.0;    // Store the volume of a box here
```

```
// box 1 specification
```

```
Box1.height = 5.0;  
Box1.length = 6.0;  
Box1.breadth = 7.0;
```

```
// box 2 specification
```

```
Box2.height = 10.0;  
Box2.length = 12.0;  
Box2.breadth = 13.0;
```

```
// volume of box 1
```

```
volume = Box1.height * Box1.length * Box1.breadth;  
cout << "Volume of Box1 : " << volume << endl;
```

```
// volume of box 2
```

```
volume = Box2.height * Box2.length * Box2.breadth;  
cout << "Volume of Box2 : " << volume << endl;
```

```
Volume of Box1 : 210
```

```
Volume of Box2 : 1560
```



# Pointer to Classes

- [https://www.tutorialspoint.com/cplusplus/cpp\\_pointer\\_to\\_class.htm](https://www.tutorialspoint.com/cplusplus/cpp_pointer_to_class.htm)
- `ptr_name->class_member;`

```
class Box {  
    public:  
        // Constructor definition  
        Box(double l = 2.0, double b = 2.0, double h = 2.0) {  
            cout <<"Constructor called." << endl;  
            length = l;  
            breadth = b;  
            height = h;  
        }  
        double Volume() {  
            return length * breadth * height;  
        }  
    private:  
        double length;    // Length of a box  
        double breadth;   // Breadth of a box  
        double height;    // Height of a box  
};
```

```
Box Box1(3.3, 1.2, 1.5);    // Declare box1  
Box Box2(8.5, 6.0, 2.0);    // Declare box2  
Box *ptrBox;                // Declare pointer to a class.  
  
// Save the address of first object  
ptrBox = &Box1;  
  
// Now try to access a member using member access operator  
cout << "Volume of Box1: " << ptrBox->Volume() << endl;  
  
// Save the address of second object  
ptrBox = &Box2;  
  
// Now try to access a member using member access operator  
cout << "Volume of Box2: " << ptrBox->Volume() << endl;
```

```
Constructor called.  
Constructor called.  
Volume of Box1: 5.94  
Volume of Box2: 102
```

# PA1 Introduction

- Problem Statement
- Input/Output Specification
- Submission
- Evaluation

# Problem Statement

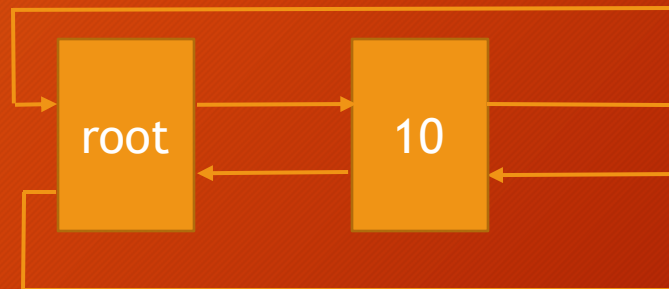
- This programming assignment asks you to read in a series of commands e.x.(PUSH 10, PUSH 9, PUSH8, .....), and execute it using stack.

Stack

PUSH 10



Data structure



# Problem Statement

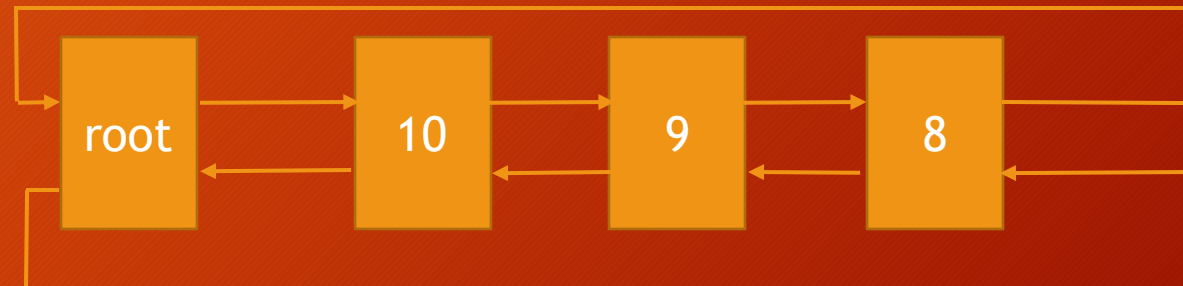
- This programming assignment asks you to read in a series of commands e.x.(PUSH 10, PUSH 9, PUSH8, .....), and execute it using stack.

Stack

```
PUSH 10  
PUSH 9  
PUSH 8
```



Data structure





# Problem Statement

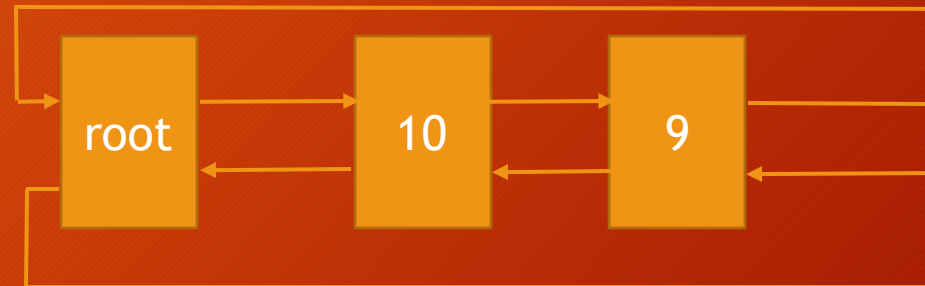
- This programming assignment asks you to read in a series of commands e.x.(PUSH 10, PUSH 9, PUSH8, .....), and execute it using stack.

Stack

```
PUSH 10  
PUSH 9  
PUSH 8  
POP
```



Data structure



# Problem Statement

- You should only fill in the TODO in stack.py and stack.cpp.

## stack.py

```
def pop(self):
    if self.num_element == 0:
        raise ValueError('Can not execute pop() on
    else:
        self.num_element -= 1
        # ---TODO:
        # Connect the second last element >> root
        # Connect root >> the second last element
        # ---
def push(self, node):
    self.num_element += 1
    # ---TODO:
    # Connect the last element >> inserted node
    # Connect the inserted node >> root
    # ---
```

## stack.cpp

```
void pop() {
    if(_num_element==0){
        throw invalid_argument( "Can not execute pop
    }
    else{
        _num_element -= 1;
        // ---TODO:
        // Connect the second last element >> root
        // Connect root >> the second last element
        // ---
    }
}

void push(MyNode* node) {
    _num_element += 1;
    // ---TODO:
    // Connect the last element >> inserted node
    // Connect the inserted node >> root
    // ---
}
```

# Notice

- For stack.py, please use python3.7 to run, while the cpp version for stack.cpp is not limited.
- You should not modify the codes which are not specified by TODO.
- You should call the class node() when storing your data structure. That is, your data structure is limited to linked lists.
- You should be able to implement stacks with  $O(1)$ .
- For stack.cpp, you should avoid memory leak.

# Input/Output Specification

input

```
1 PUSH 1
2 PUSH 2
3 POP
4 POP
5 PUSH 3
6 POP
7 PUSH 4
8 POP
9 PUSH 5
10 POP
```

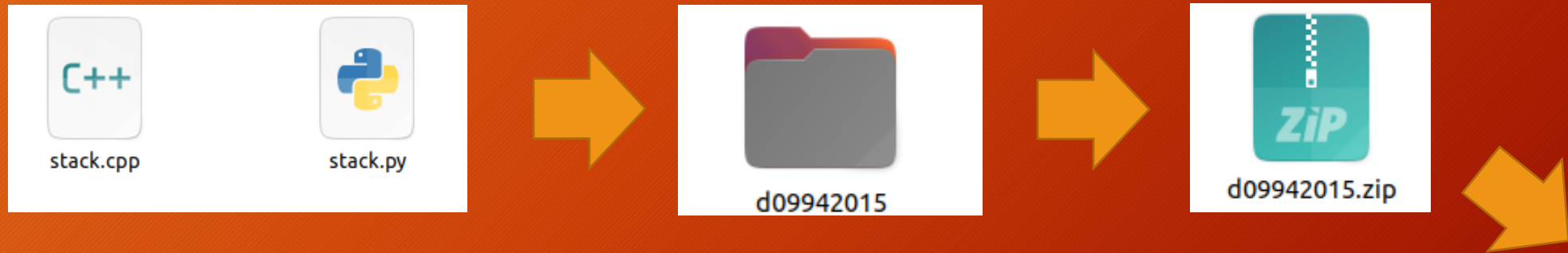
output

```
1 >>Node1
2 >>Node1>>Node2
3 >>Node1
4
5 >>Node3
6
7 >>Node4
8
9 >>Node5
10
```



# Submission

- Due on 4/1, at 4:00 am
- Please put stack.py and stack.cpp into a directory named studentID and compress the directory into studentID.zip, and then upload this file to ceiba



作業區									
作業列表									
已指派的作業									
名稱	成員	繳文方法	成績比重	繳文期限	逾期繳文	繳文日期	作業評語	作業視摩	
hw1_written	個人	線上繳交	3	2021-03-25 09	不可以	--	--	<a href="#">進入</a>	

# Evaluation

- All of our test cases will not execute `pop()` on an empty stack. You won't need to handle this exception.

in our testcase

```
PUSH 10  
PUSH 9  
PUSH 8  
POP
```

not in our testcase

```
PUSH 10  
POP  
POP  
POP
```

# Evaluation

- Correctness and Time Complexity : 80%
- Memory Management : 20%

# Evaluation (Correctness and Time Complexity)

- We will evaluate your code on five test cases.

Input :

input\_1.txt

input\_2.txt

input\_3.txt

input\_4.txt

input\_5.txt

```
1 PUSH 1
2 PUSH 2
3 POP
4 POP
5 PUSH 3
6 POP
7 PUSH 4
8 POP
9 PUSH 5
10 POP
```

Golden output:

golden\_1.txt

golden\_2.txt

golden\_3.txt

golden\_4.txt

golden\_5.txt

```
1 >>Node1
2 >>Node1>>Node2
3 >>Node1
4
5 >>Node3
6
7 >>Node4
8
9 >>Node5
10
```



# Evaluation (Correctness and Time Complexity)

- Command line argument for correctness:
  - `python stack.py --input input_1.txt --output output_py_1.txt`
  - `g++ -g -o stack stack.cpp; ./stack input_1.txt input_2.txt`
- Command line argument for time complexity:
  - `python stack.py --input input_1.txt`
  - `g++ -g -o stack stack.cpp; ./stack input_1.txt`

# Evaluation (Correctness and Time Complexity)

- Evaluation script : evaluation.sh
- Command for evaluation.sh:
  - `bash evaluation.sh`

python & cpp are incorrect

```
==evaluating correctness==  
stack.py is incorrect in test case: input_1.txt  
stack.cpp is incorrect in test case: input_1.txt  
stack.py is incorrect in test case: input_2.txt  
stack.cpp is incorrect in test case: input_2.txt  
stack.py is incorrect in test case: input_3.txt  
stack.cpp is incorrect in test case: input_3.txt
```

python & cpp are correct

```
==evaluating correctness==  
stack.py is correct in test case: input_1.txt  
stack.cpp is correct in test case: input_1.txt  
stack.py is correct in test case: input_2.txt  
stack.cpp is correct in test case: input_2.txt  
stack.py is correct in test case: input_3.txt  
stack.cpp is correct in test case: input_3.txt
```

# Evaluation (Correctness and Time Complexity)

- Time Complexity:
  - **You should manually check the time complexity**
  - input\_2.txt: 100 commands, input\_3.txt: 10,000 commands
  - Python runtime of input\_3.txt: 0.00992s ~  $0.00019s \cdot 100 = 0.019s$
  - CPP runtime of input\_3.txt: 0.00427s ~  $0.00007s \cdot 100 = 0.007s$

==evaluating runtime==

```
stack.py run time of input_1.txt: 0.00012s
stack.cpp run time of input_1.txt: 0.00004s
stack.py run time of input_2.txt: 0.00019s
stack.cpp run time of input_2.txt: 0.00007s
stack.py run time of input_3.txt: 0.00992s
stack.cpp run time of input_3.txt: 0.00427s
```



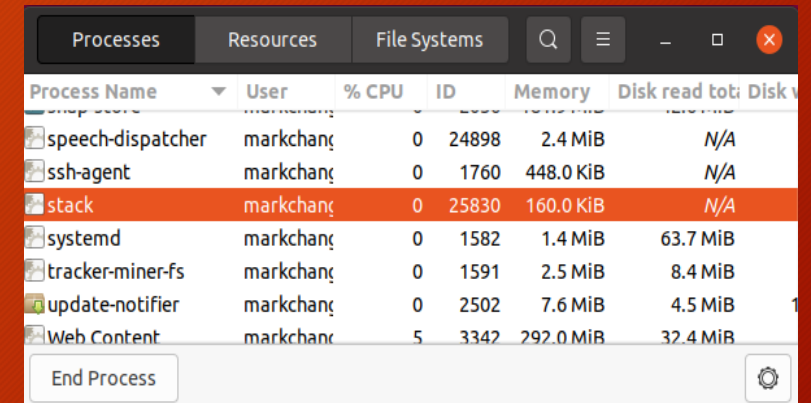
# Evaluation (Memory Management)

- Command line argument for memory management:

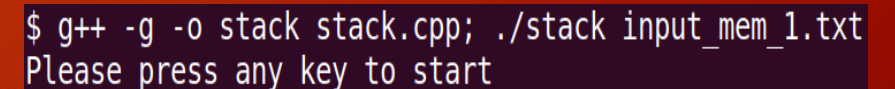
- `g++ -g -o stack stack.cpp; ./stack input_mem_1.txt`

- Steps:

- 1. enter the above command
  - 2. open "System Monitor", search for "stack"
  - 3. press any key on terminal to start "stack"
  - 4. open "System Monitor", search for "stack"
  - 5. press any key on terminal to end "stack"



Process Name	User	% CPU	ID	Memory	Disk read tot	Disk w
speech-dispatcher	markchang	0	24898	2.4 MiB	N/A	
ssh-agent	markchang	0	1760	448.0 KiB	N/A	
stack	markchang	0	25830	160.0 KiB	N/A	
systemd	markchang	0	1582	1.4 MiB	63.7 MiB	
tracker-miner-fs	markchang	0	1591	2.5 MiB	8.4 MiB	
update-notifier	markchang	0	2502	7.6 MiB	4.5 MiB	
Web Content	markchang	5	3342	292.0 MiB	32.4 MiB	



```
$ g++ -g -o stack stack.cpp; ./stack input_mem_1.txt
Please press any key to start
```



# Evaluation (Memory Management)

- Memory Leak

Before press key to start

Processes						
Resources						
File Systems						
Process Name	User	% CPU	ID	Memory	Disk read tot	Disk v
speech-dispatcher	markchan	0	24898	2.4 MiB	N/A	
ssh-agent	markchan	0	1760	448.0 KiB	N/A	
stack	markchan	0	25830	160.0 KiB	N/A	
systemd	markchan	0	1582	1.4 MiB	63.7 MiB	
tracker-miner-fs	markchan	0	1591	2.5 MiB	8.4 MiB	
update-notifier	markchan	0	2502	7.6 MiB	4.5 MiB	1
Web Content	markchan	5	3342	292.0 MiB	32.4 MiB	

After press key to start

Processes						
Resources						
File Systems						
Process Name	User	% CPU	ID	Memory	Disk read tot	Disk v
speech-dispatcher	markchan	0	24898	2.4 MiB	N/A	
ssh-agent	markchan	0	1760	448.0 KiB	N/A	
stack	markchan	0	25863	15.2 MiB	N/A	
systemd	markchan	0	1582	1.4 MiB	63.7 MiB	
tracker-miner-fs	markchan	0	1591	2.5 MiB	8.4 MiB	
tracker-store	markchan	0	25852	11.0 MiB	N/A	
update-notifier	markchan	0	2502	7.6 MiB	4.5 MiB	1

# Evaluation (Memory Management)

- No Memory Leak

Before press key to start

Processes						
Resources						
File Systems						
Process Name	User	% CPU	ID	Memory	Disk read tot	Disk
speech-dispatcher	markchan	0	24898	2.4 MiB	N/A	
ssh-agent	markchan	0	1760	448.0 KiB	N/A	
stack	markchan	0	25830	160.0 KiB	N/A	
systemd	markchan	0	1582	1.4 MiB	63.7 MiB	
tracker-miner-fs	markchan	0	1591	2.5 MiB	8.4 MiB	
update-notifier	markchan	0	2502	7.6 MiB	4.5 MiB	
Web Content	markchan	5	3342	292.0 MiB	32.4 MiB	

## After press key to start

Process Name	User	% CPU	ID	Memory	Disk read total	Disk write total
speech-dispatcher	markchang	0	24898	2.4 MiB	N/A	N/A
ssh-agent	markchang	0	1760	448.0 KiB	N/A	N/A
stack	markchang	0	25830	160.0 KiB	N/A	N/A
systemd	markchang	0	1582	1.4 MiB	63.7 MiB	0.0 MiB
tracker-miner-fs	markchang	0	1591	2.5 MiB	8.4 MiB	0.0 MiB
update-notifier	markchang	0	2502	7.6 MiB	4.5 MiB	0.0 MiB
Web Content	markchang	5	3342	293.5 MiB	32.4 MiB	0.0 MiB

End Process

# Scoring Criteria

## 1. Correctness and Time Complexity : 80%

	input_1.txt 20 lines PUSH only	input_2.txt 100 lines PUSH & POP	input_3.txt 10,000 lines PUSH & POP	input_4.txt 2,000 lines PUSH only	input_5.txt 50,000 lines PUSH & POP
stack.py	8%	8%	8%	8%	8%
stack.cpp	8%	8%	8%	8%	8%

## 2. Memory Management : 20%

	input_mem_1.txt 1,000,000 lines PUSH & POP	input_mem_2.txt 5,000,000 lines PUSH & POP
stack.cpp	10%	10%

# Any Questions?

- If you have any questions, feel free to contact TA.
  - 張富傑(Mark)
  - email : [d09942015@ntu.edu.tw](mailto:d09942015@ntu.edu.tw)
  - office hour: 週四(17:30~18:20)at博理530
  - phone : 0989922753