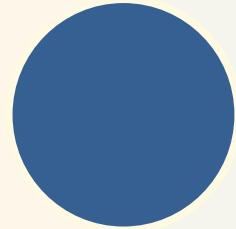
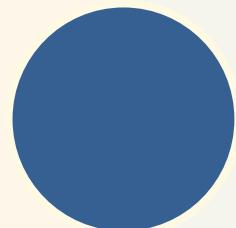




# Agenda



Data Structures : Linked-list, Queue, Stack



Structures, Unions, Typedef, Enumerations  
File in C++



# DATA STRUCTURES

โครงสร้างข้อมูล



# DATA STRUCTURES

## โครงสร้างข้อมูล

- เป็นวิธีการในการจัดเก็บและจัดระเบียบข้อมูลของคอมพิวเตอร์
- เพื่อให้เข้าถึงและแก้ไขข้อมูลได้อย่างมีประสิทธิภาพ
- การเลือกโครงสร้างข้อมูล จะขึ้นกับลักษณะของการใช้งานข้อมูล ปริมาณข้อมูลที่จะจัดเก็บ และความถี่ในการใช้งาน
- การเลือกโครงสร้างข้อมูลที่เหมาะสม จะทำให้ใช้พื้นที่ใน RAM เกิดประโยชน์สูงสุด
- โครงสร้างข้อมูลมีบทบาทอย่างมากในการจัดการชุดข้อมูลขนาดใหญ่ การสร้างดัชนีข้อมูล เส้นทางในการนำทาง algorithm ฯลฯ



## ① PRIMITIVE DATA STRUCTURE

โครงสร้างข้อมูลชนิดพื้นฐาน

- โครงสร้างข้อมูลหลักที่ทำงานโดยตรงกับคำสั่งระดับคอมพิวเตอร์
- ใช้กับการดึงและเปลี่ยนแปลงค่าข้อมูลชนิดต่าง ๆ
- มีประสิทธิภาพกับข้อมูลที่มีขนาดเล็ก

**integer**

จำนวนเต็ม

**float**

จำนวนจริง

**character**

ตัวอักษร

**boolean**

ตรรกะ



## ② NON-PRIMATIVE DATA STRUCTURE

โครงสร้างข้อมูลชนิดไม่ใช้พื้นฐาน

- ซับซ้อนมากกว่าโครงสร้างข้อมูลชนิดพื้นฐาน
- นิยมใช้เก็บข้อมูลขนาดใหญ่ที่มีความเชื่อมโยงกัน

### Linear Data Structure

โครงสร้างข้อมูลเชิงเส้น

เป็นการจัดเรียงข้อมูลในลำดับ

การเข้าถึงข้อมูลทำได้โดยเริ่มจากจุดแรกไปทีละลำดับตามโครงสร้าง

Arrays, Linked lists, Stacks, Queues

### Non-Linear Data Structure

โครงสร้างข้อมูลไม่เชิงเส้น

การจัดเรียงข้อมูลอย่างไม่มีลำดับ

แต่เน้นความสัมพันธ์แบบลำดับชั้นหรือการเชื่อมโยงที่ซับซ้อน

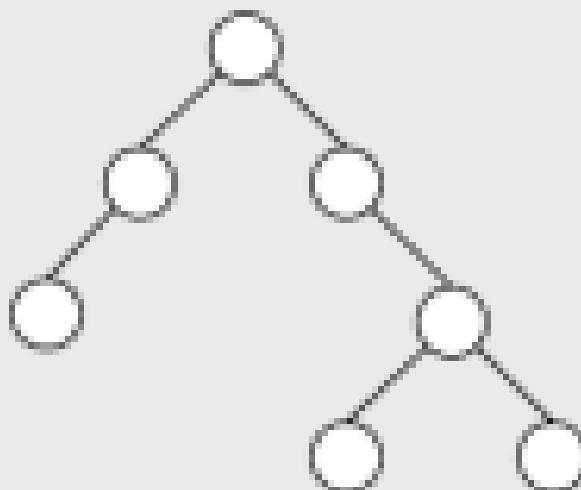
Trees, Graphs

## Non-Linear Data Structure

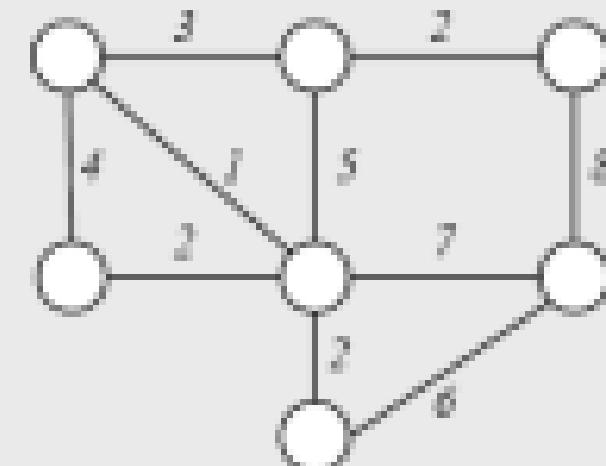
โครงสร้างข้อมูลไม่เชิงเส้น

การจัดเรียงข้อมูลอย่างไม่มีลำดับ  
แต่เน้นความสัมพันธ์แบบลำดับชั้นหรือการเชื่อมโยงที่ซับซ้อน

Trees, Graphs



Tree



Graph



## Linear Data Structure

โครงสร้างข้อมูลเชิงเส้น

เป็นการจัดเรียงข้อมูลในลำดับ  
การเข้าถึงข้อมูลทำได้โดยเริ่มจากจุดแรกไปทีละลำดับตามโครงสร้าง

- **Arrays**
- **Linked lists**
- **Stacks**
- **Queues**



## Static Data Structure

โครงสร้างข้อมูลแบบคงที่

โครงสร้างและขนาดถูกระบุตایตัว  
ไม่สามารถเปลี่ยนแปลงหลังการสร้าง  
เช่น Array

## Dynamic Data Structure

โครงสร้างข้อมูลแบบเปลี่ยนแปลงได้

โครงสร้างที่สามารถปรับเปลี่ยนขนาดและ  
รูปแบบได้ระหว่างการทำงาน  
เช่น Linked Lists



## DATA STRUCTURE

# linked-list



# Linked-list

เป็นโครงสร้างข้อมูลแบบ Linear ชุดข้อมูลที่ถูกจัดเก็บภายในหน่วยที่เรียกว่า node  
แต่ละ node ประกอบด้วยสองส่วนหลัก  
คือ ข้อมูลที่เราต้องการเก็บ และส่วนอ้างอิง (ลิงก์) ที่ซึ่งไปยัง node ถัดไป

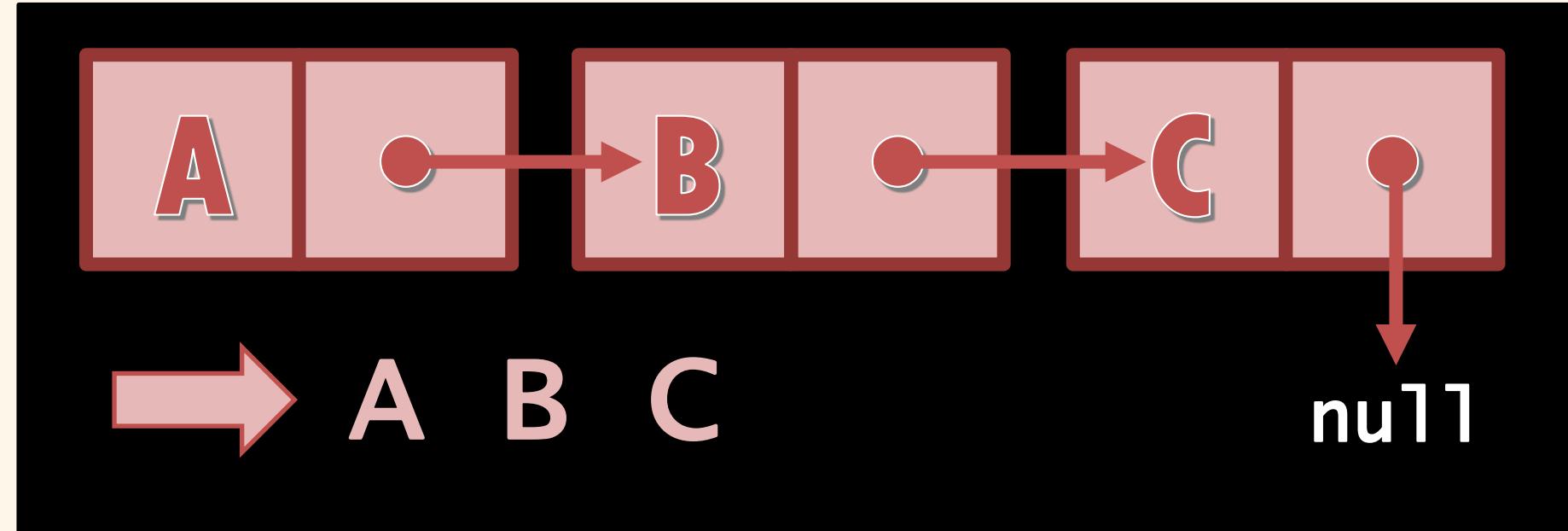




## Array

grade [“A”, “B”, “C”] << ประกาศเก็บ array 3 ช่อง

## Linked-list



## Singly linked-list

<< สำรวจข้อมูลได้ทิศทางเดียว



```
#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node *next;

    Node(int data) {
        this->data = data;
        this->next = nullptr;
    }

};

void printList(Node *n) {
    while (n != nullptr) {
        cout << n->data << " ";
        n = n-> next;
    }
}
```

```
int main() {
    Node *head = new Node(1);
    Node *second = new Node(2);
    Node *third = new Node(3);

    head->next = second;
    second->next = third;

    printList(head);

    return 0;
}
```

The terminal window shows the command `C:\WINDOWS\system32` followed by the output of the program: `1 2 3`. The path `c:\Users\Peerapat\posn67\01>` is also visible at the bottom.



```
#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node *next;

    Node(int data) {
        this->data = data;
        this->next = nullptr;
    }

};

void printList(Node *n) {
    while (n != nullptr) {
        cout << n->data << " ";
        n = n-> next;
    }
}
```

```
int main() {
    Node *head = new Node(1);
    Node *second = new Node(2);
    Node *third = new Node(3);
```

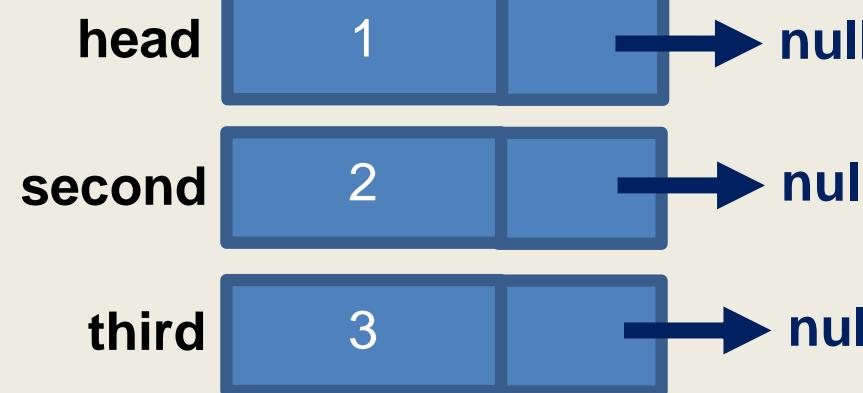
constructor : node

**data**      **next** → **nullptr**

```
C:\WINDOWS\system32 + 
1 2 3
c:\Users\Peerapat\posn67\01>
```



# Linked-list



```
Node(int data) {
    this->data = data;
    this->next = nullptr;
}

void printList(Node *n) {
    while (n != nullptr) {
        cout << n->data << " ";
        n = n->next;
    }
}
```

```
int main() {
    Node *head = new Node(1);
    Node *second = new Node(2);
    Node *third = new Node(3);

    head->next = second;
    second->next = third;

    printList(head);

    return 0;
}
```

The terminal window shows the output of the program, which prints the values 1, 2, and 3, indicating that the linked list has been successfully created and printed.

```
C:\WINDOWS\system32 + ▾ - □ ×
1 2 3
c:\Users\Peerapat\posn67\01>
```



# Linked-list

```
#include <iostream>
using namespace std;
```

```
class Node {
public:
    int data;
    Node *next;
    Node(int i) {
        this->data = i;
        this->next = nullptr;
    }
};
```

~~head > null~~  
**head -> second**

```
void printList(Node *n) {
    while (n != nullptr) {
        cout << n->data << " ";
        n = n->next;
    }
}
```

```
int main() {
    Node *head = new Node(1);
    Node *second = new Node(2);
    Node *third = new Node(3);

    head->next = second;
    second->next = third;

    printList(head);

    return 0;
}
```

```
C:\WINDOWS\system3 + □ ×
1 2 3
c:\Users\Peerapat\posn67\01>
```



# Linked-list

```
#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node *next;

    Node(int i) {
        this->data = i;
        this->next = nullptr;
    }
};

void printList(Node *n) {
    while (n != nullptr) {
        cout << n->data << " ";
        n = n->next;
    }
}
```

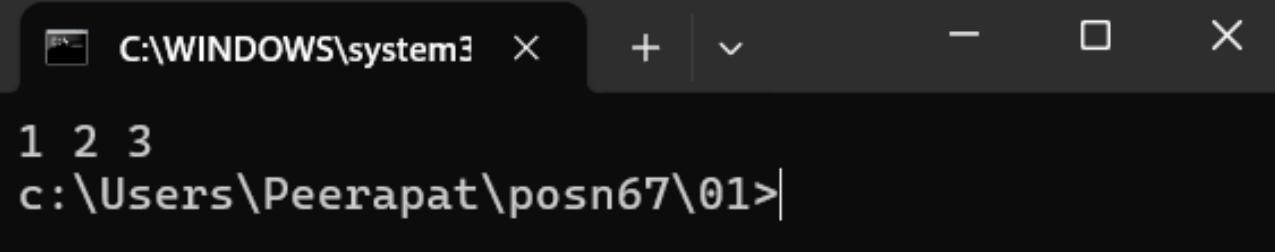
~~second -> null~~  
**second -> third**

```
int main() {
    Node *head = new Node(1);
    Node *second = new Node(2);
    Node *third = new Node(3);

    head->next = second;
    second->next = third;

    printList(head);

    return 0;
}
```



The terminal window shows the output of the program. It displays three integers: 1, 2, and 3, separated by spaces. The path to the terminal window is shown at the bottom: c:\Users\Peerapat\posn67\01>.



# Linked-list

```
#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node *next;

    Node(int data) {
        this->data = data;
        this->next = nullptr;
    }

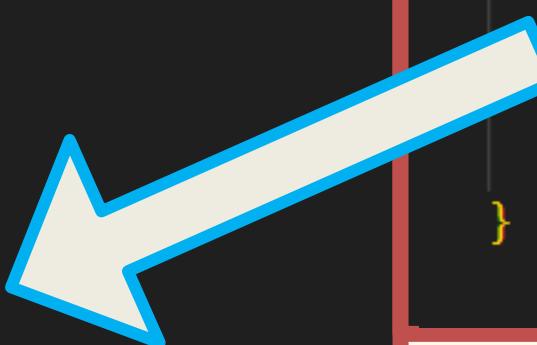
    void printList(Node *n) {
        while (n != nullptr) {
            cout << n->data << " ";
            n = n->next;
        }
    }
};
```

```
int main() {
    Node *head = new Node(1);
    Node *second = new Node(2);
    Node *third = new Node(3);

    head->next = second;
    second->next = third;

    printList(head);

    return 0;
}
```



```
C:\WINDOWS\system32 + ▾ - □ ×
1 2 3
c:\Users\Peerapat\posn67\01>
```



# Linked-list

```
#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node *next;

    Node(int data) {
        this->data = data;
        this->next = nullptr;
    }
};

void printList(Node *n) {  
    while (n != nullptr) {  
        cout << n->data << " ";  
        n = n-> next;  
    }  
}
```

while loop  
null -> break

```
int main() {
    Node *head = new Node(1);
    Node *second = new Node(2);
    Node *third = new Node(3);

    head->next = second;
    second->next = third;

    printList(head);

    return 0;
}
```

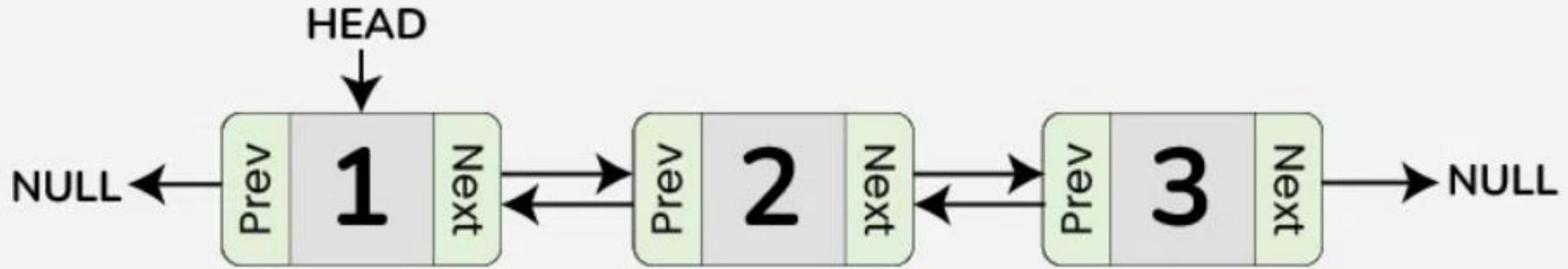
C:\WINDOWS\system32

1 2 3

c:\Users\Peerapat\posn67\01>



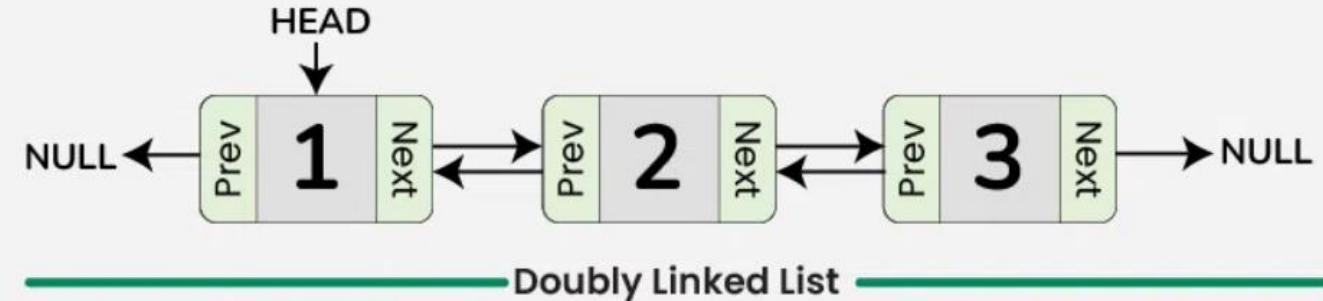
# Doubly Linked-list



Doubly Linked List



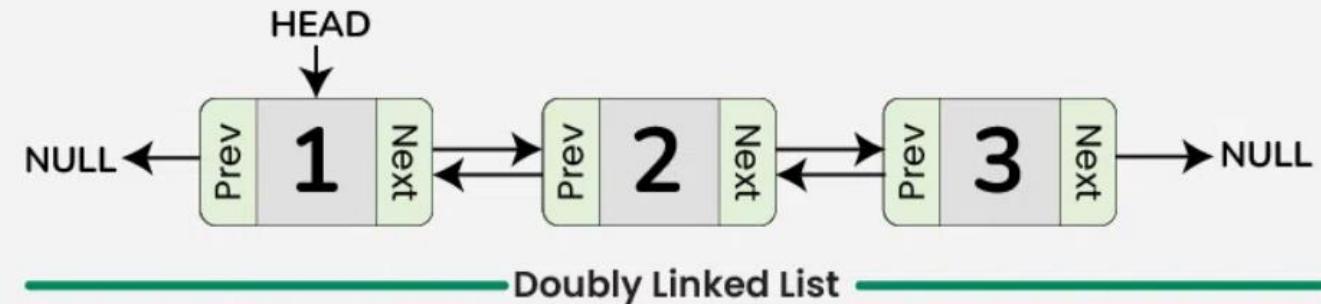
## Doubly Linked-list



```
class Node {  
public:  
    int data;  
    Node *prev;  
    Node *next;  
  
    Node(int data) {  
        this->data = data;  
        this->prev = nullptr;  
        this->next = nullptr;  
    }  
};
```



## Doubly Linked-list

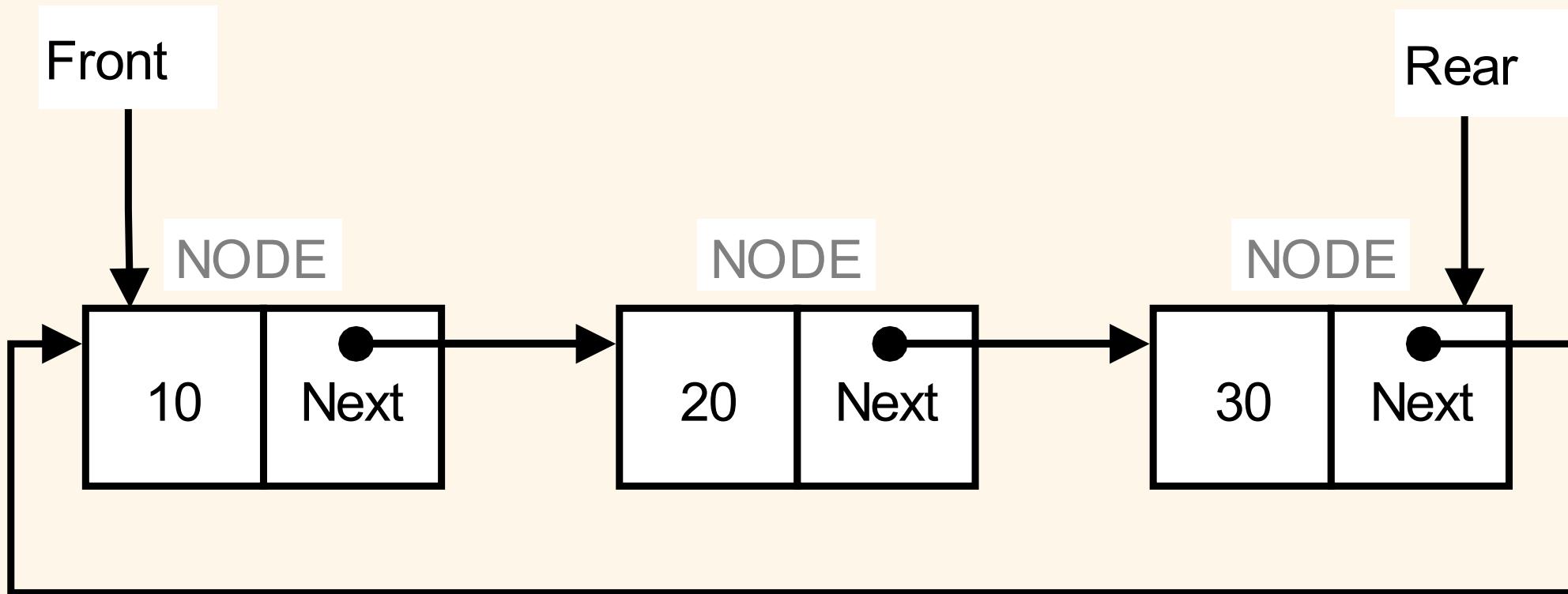


```
class Node {  
public:  
    int data;  
    Node *prev;  
    Node *next;  
  
    Node(int data) {  
        this->data = data;  
        this->prev = nullptr;  
        this->next = nullptr;  
    }  
};
```

```
int main() {  
    Node *head = new Node(1);  
    Node *second = new Node(2);  
    Node *third = new Node(3);  
  
    head->next = second;  
    second->prev = head;  
    second->next = third;  
    third->prev = second;  
  
    printList(head);  
  
    return 0;  
}
```

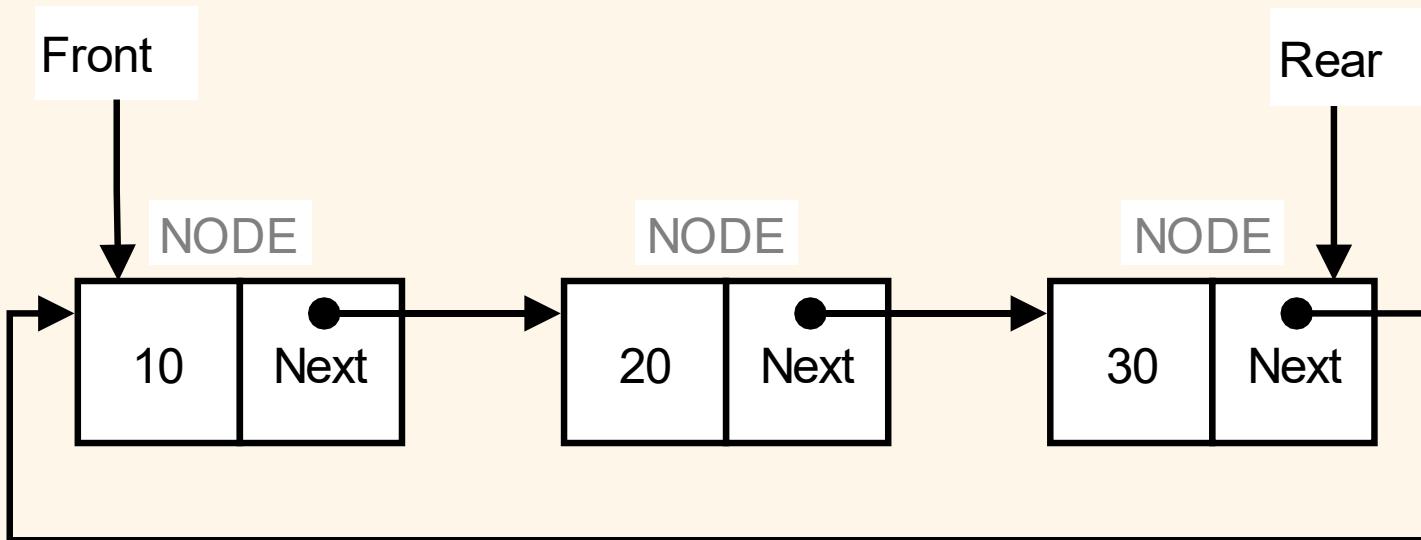


# Circular Linked-list





# Circular Linked-list



```
head->next = second;
second->next = third;
third->next = head;
```



# Linked-list



# **DATA STRUCTURE**

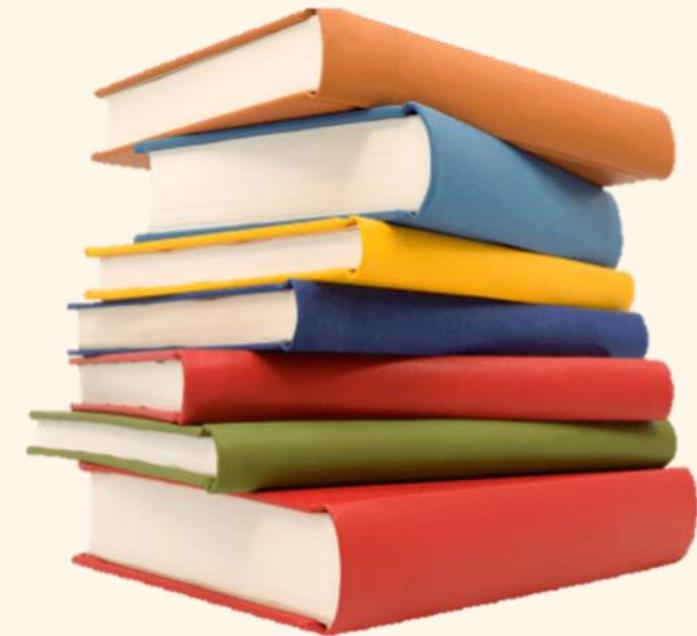
# **Queue . Stack**



# Queue



# Stack



<https://courses.grainger.illinois.edu/cs225/sp2019/notes/quacks/>

# Queue



โครงสร้างข้อมูลที่มีการเพิ่มข้อมูลเข้ามาทางด้านท้าย  
และการดึงข้อมูลออกทางด้านหน้า  
มีประโยชน์ต่อการจัดเรียงลำดับการทำงาน

**First In, First Out**

**FIFO**

(เข้าก่อนออกก่อน)



# Stack

โครงสร้างข้อมูลแบบ linear ที่มีการเพิ่มและดึงข้อมูลออกจากกับข้อมูลล่าสุดเท่านั้น

**Last In, First Out**

**LIFO**

(เข้าหลังออกก่อน)



# Queue



โครงสร้างข้อมูลที่มีการเพิ่มข้อมูลเข้ามาทางด้านท้าย  
และการดึงข้อมูลออกทางด้านหน้า  
มีประโยชน์ต่อการจัดเรียงลำดับการทำงาน

**First In, First Out**

**FIFO**

(เข้าก่อนออกก่อน)



arr 0 1 2 3 4

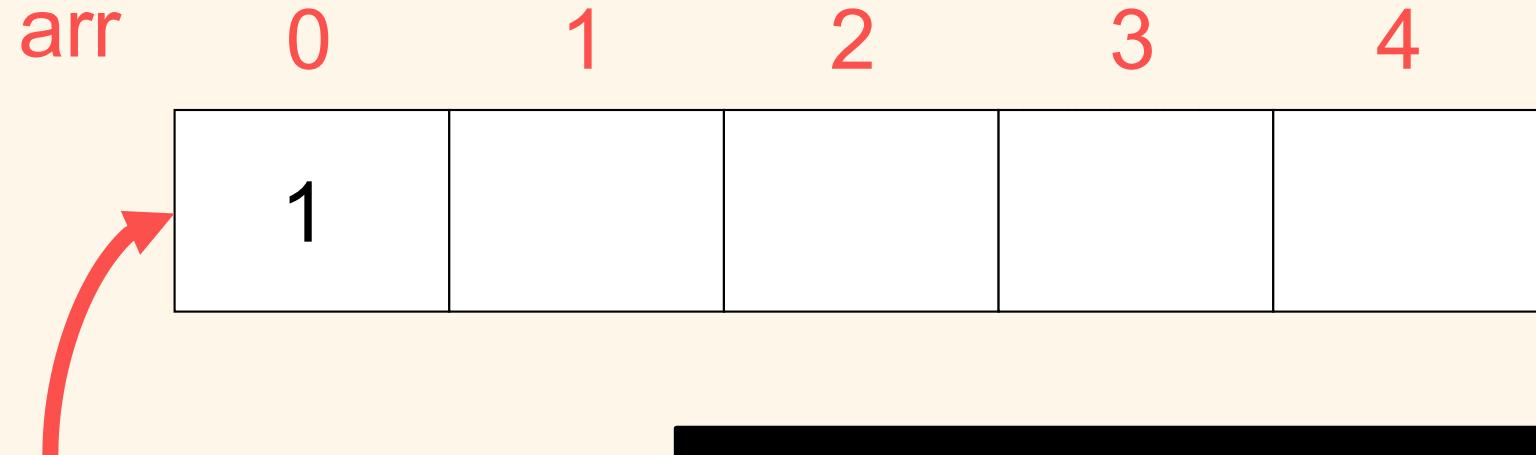
index  
↑



front = -1

rear = -1

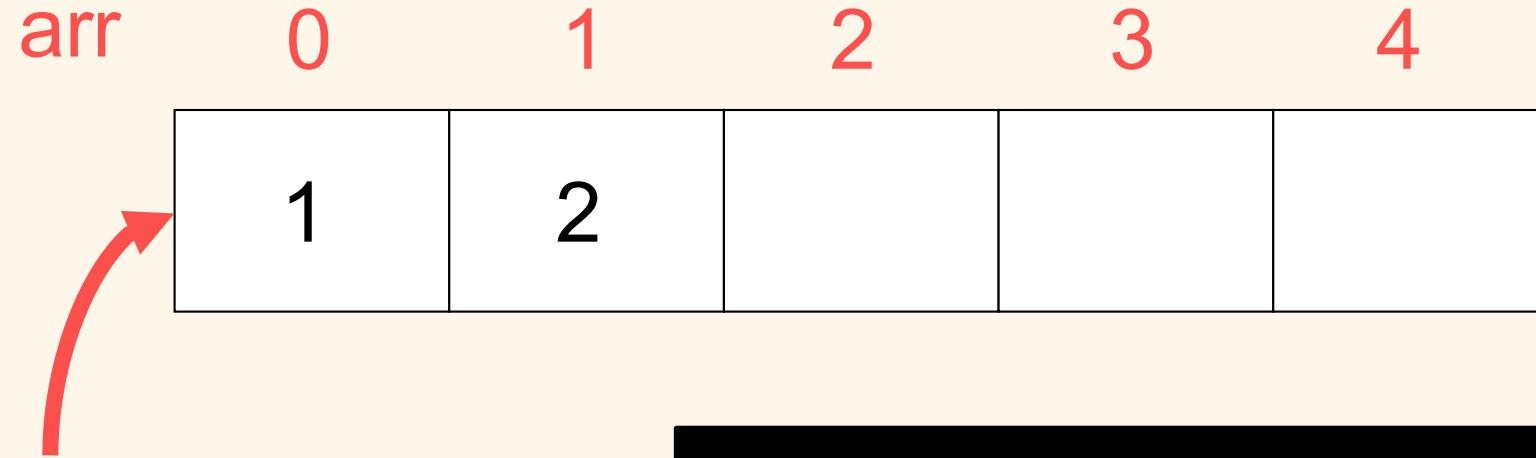
arr[front]  
arr[rear]



front = 0

rear = 0

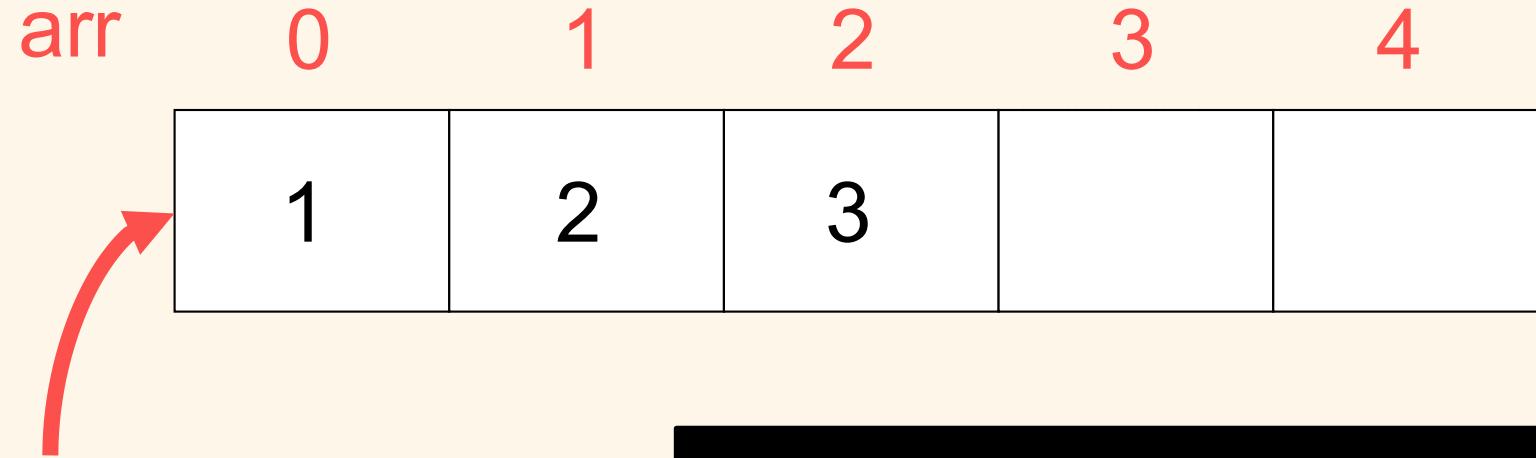
arr[front]  
arr[rear]



front = 0

rear = 1

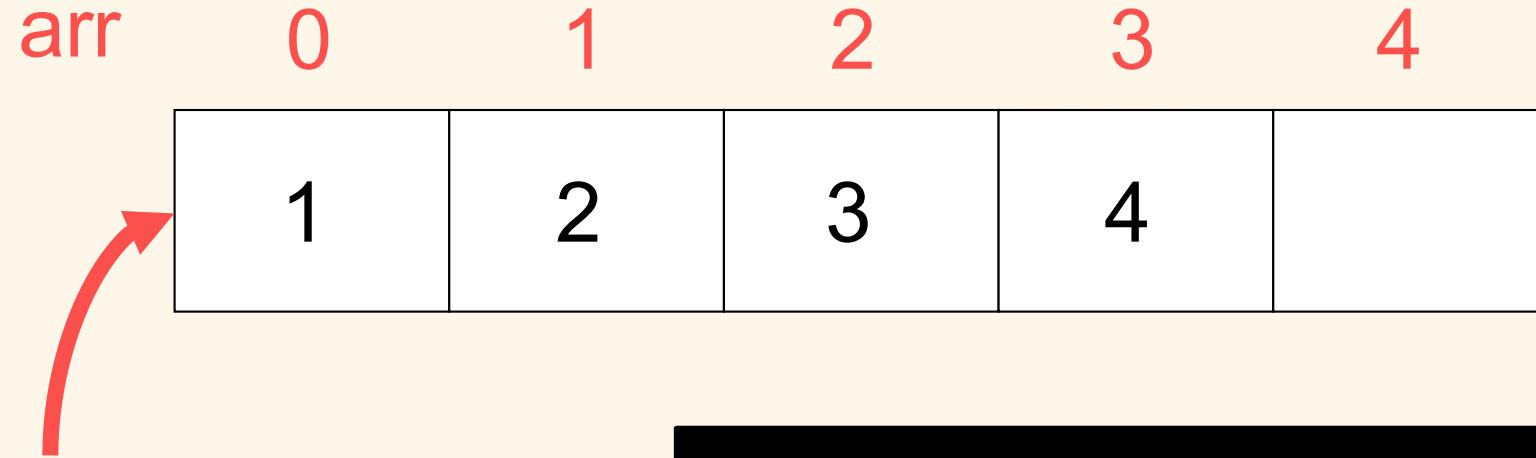
arr[front]  
arr[rear]



front = 0

rear = ~~0~~ \* 2

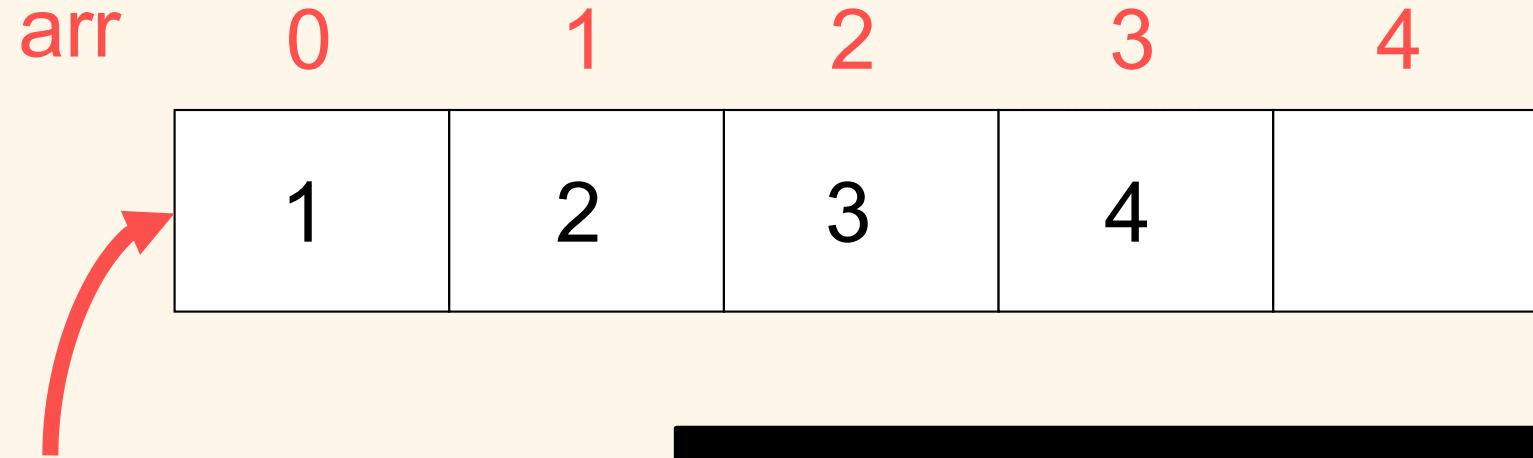
arr[front]  
arr[rear]



front = 0

rear = ✕ ✕ ✕ 3

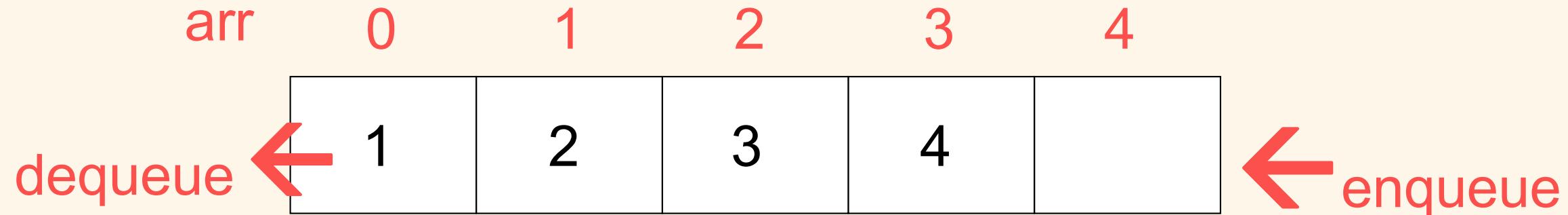
arr[front]  
arr[rear]



front = 0

rear = ✕ ✕ ✕ 3

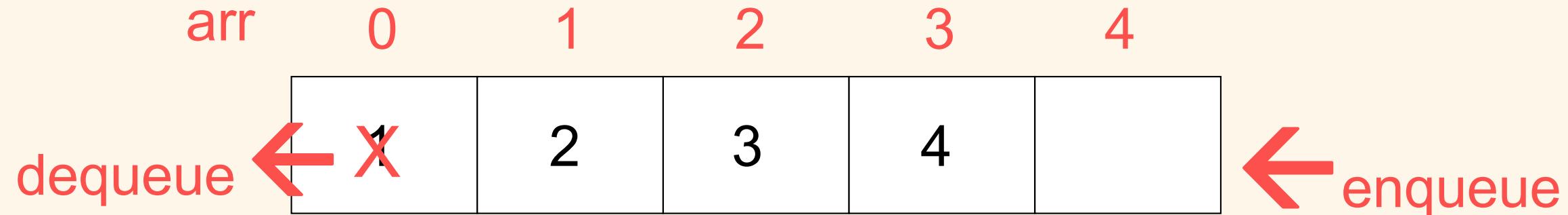
arr[front] → front  
arr[rear] → rear, last



front = 0

rear = ~~0~~ ~~1~~ ~~2~~ 3

arr[front] → front  
arr[rear] → rear, last



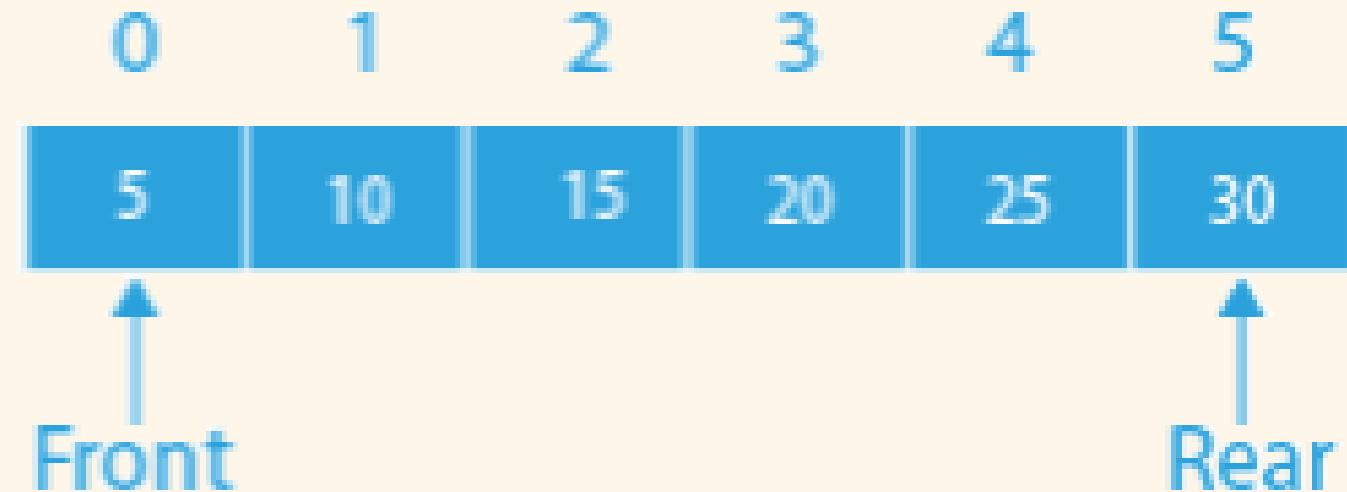
front = 0

rear = 3

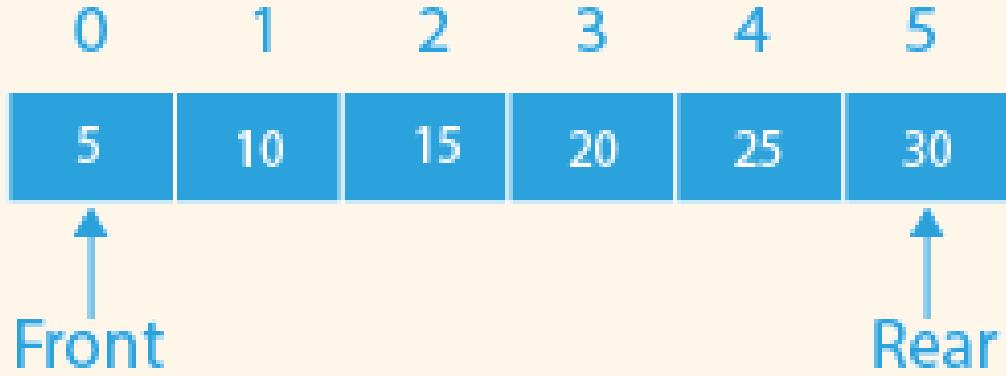
arr[front] → front  
arr[rear] → rear, last



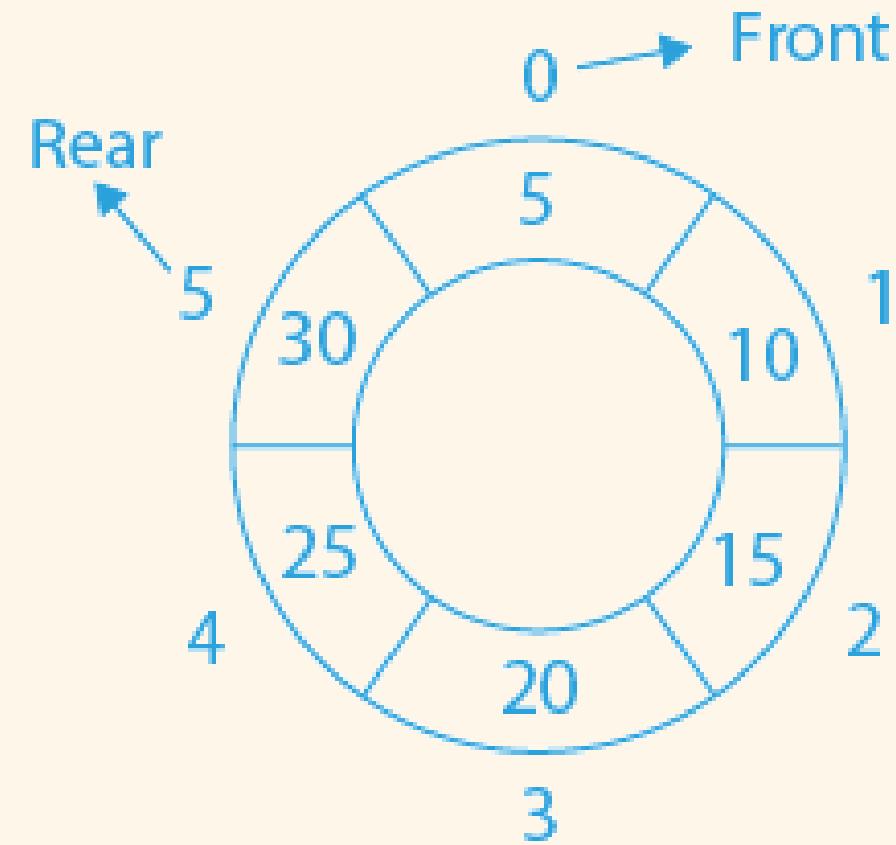
1. Linear Queue เป็น Queue ที่ข้อมูลจะถูกเพิ่ม (enqueue) เข้าไปทางด้านหลัง และลบ (dequeue) ออกจากด้านหน้า



Linear Queue



Linear Queue

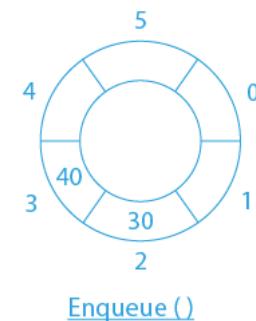
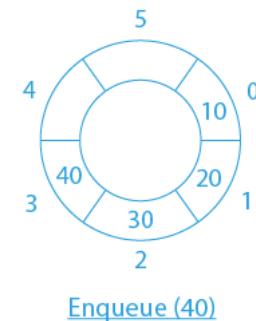
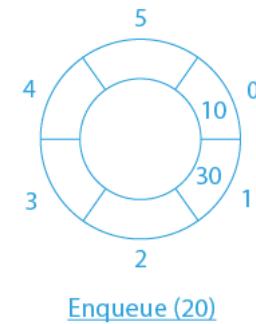
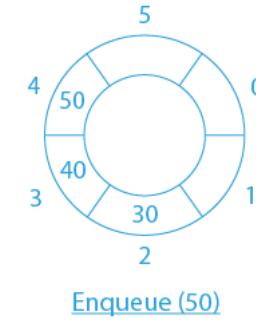
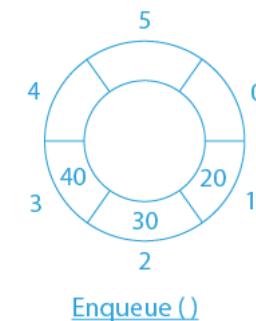
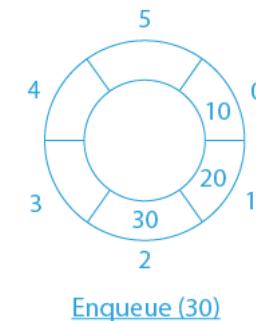
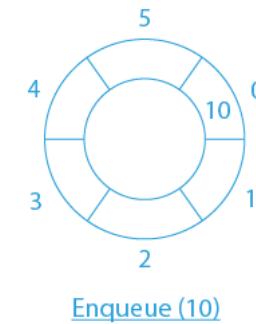


Circular Queue



2. Circular Queue Queue ที่เพิ่มประสิทธิภาพ  
โดยการเชื่อมต่อแบบ环形队列 (Circular Queue)

เป็นการแก้ปัญหาพื้นที่เหลือทิ้งโดยการนำพื้นที่ที่  
เกิดจากการลบข้อมูลกลับมาใช้ใหม่ได้



Queue

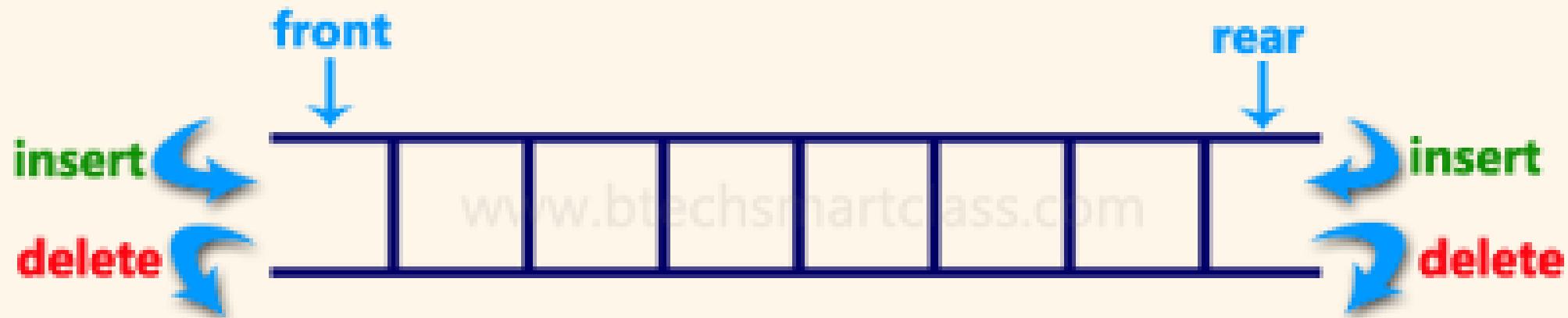


3. Priority Queue abstract data type ที่แต่ละข้อมูลจะมีค่าความสำคัญ (priority) กำกับไว้ การลบข้อมูล (dequeue) จะลบข้อมูลที่มีค่าความสำคัญสูงสุดออกไปก่อน

Priority	3	2	3	2	1		
Element	10	20	30	40	50		
Index	0	1	2	3	4	5	6



4. Double-Ended Queue (Deque) Queue ที่มีความยืดหยุ่นในการเพิ่ม (insert)  
หรือลบ (delete) ข้อมูลได้ทั้งจากด้านหน้าและด้านหลัง





# สามารถเจอ Queue ได้จากไหน?

Brother HL-2040 series on PAVILIONWIN7 - Paused							
Printer	Document	View	Status	Owner	Pages	Size	Submitted
	Microsoft Word - Renovation email.doc		HomeGroupUser\$	HomeGroupUser\$	4	1.08 MB	9:34:27 PM 4/27/2009
	Microsoft Word - Test2.docx		HomeGroupUser\$	HomeGroupUser\$	1	1.63 KB	9:32:55 PM 4/27/2009
	Microsoft Word - Mutual of America inquiry.doc		Woody	Woody	1	13.7 KB	9:32:32 PM 4/27/2009
	Microsoft Word - Allow Bongkod to pick up Ad...		Woody	Woody	2	5.05 MB	9:31:24 PM 4/27/2009

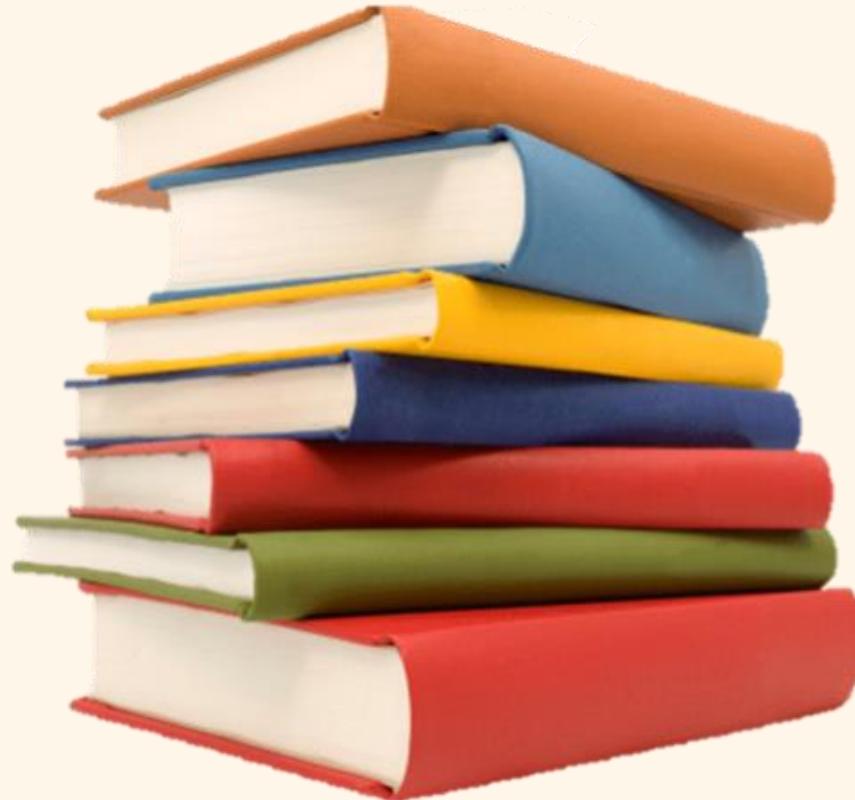
4 document(s) in queue



# Queue Library

## std::queue

- **push(data)** เพิ่มข้อมูล 'data' เข้าไปที่ด้านหลังสุดของ Queue
- **pop()** ลบข้อมูลตัวแรกออกจาก Queue
- **front()** คืนค่าแบบอ้างอิงไปยังข้อมูลตัวแรกใน Queue
- **back()** คืนค่าแบบอ้างอิงไปยังข้อมูลตัวสุดท้ายใน Queue
- **empty()** ตรวจสอบว่า Queue ว่างอยู่หรือไม่
- **size()** คืนค่าจำนวนข้อมูลทั้งหมดใน Queue



# Stack

โครงสร้างข้อมูลแบบ linear ที่มีการเพิ่มและดึงข้อมูลออกจากกับข้อมูลล่าสุดเท่านั้น

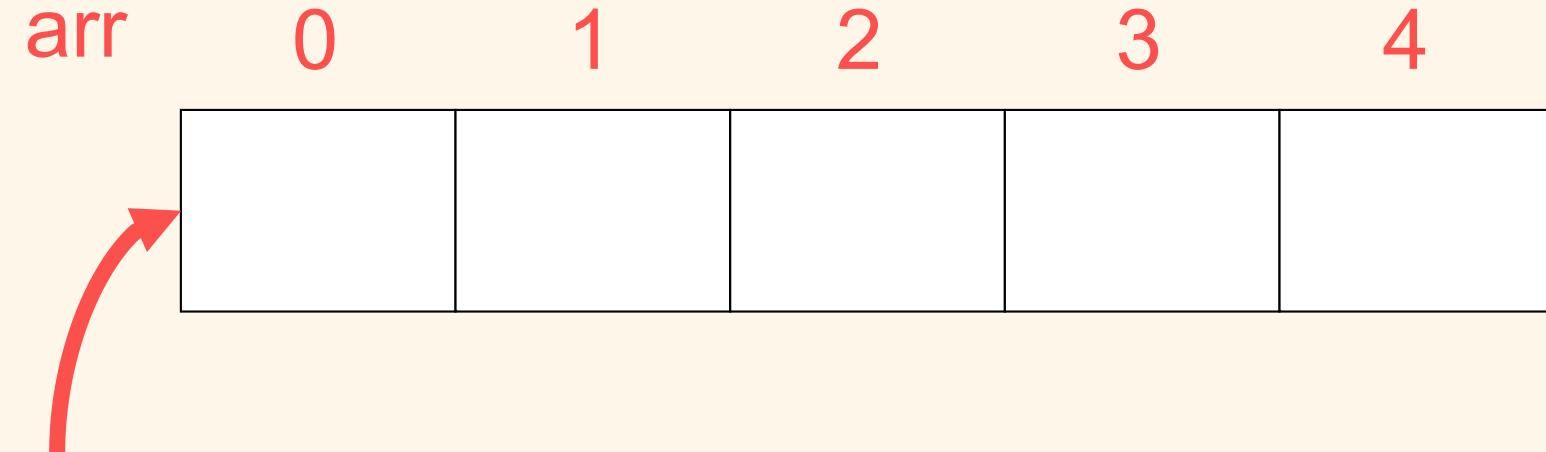
**Last In, First Out**

**LIFO**

(เข้าหลังออกก่อน)



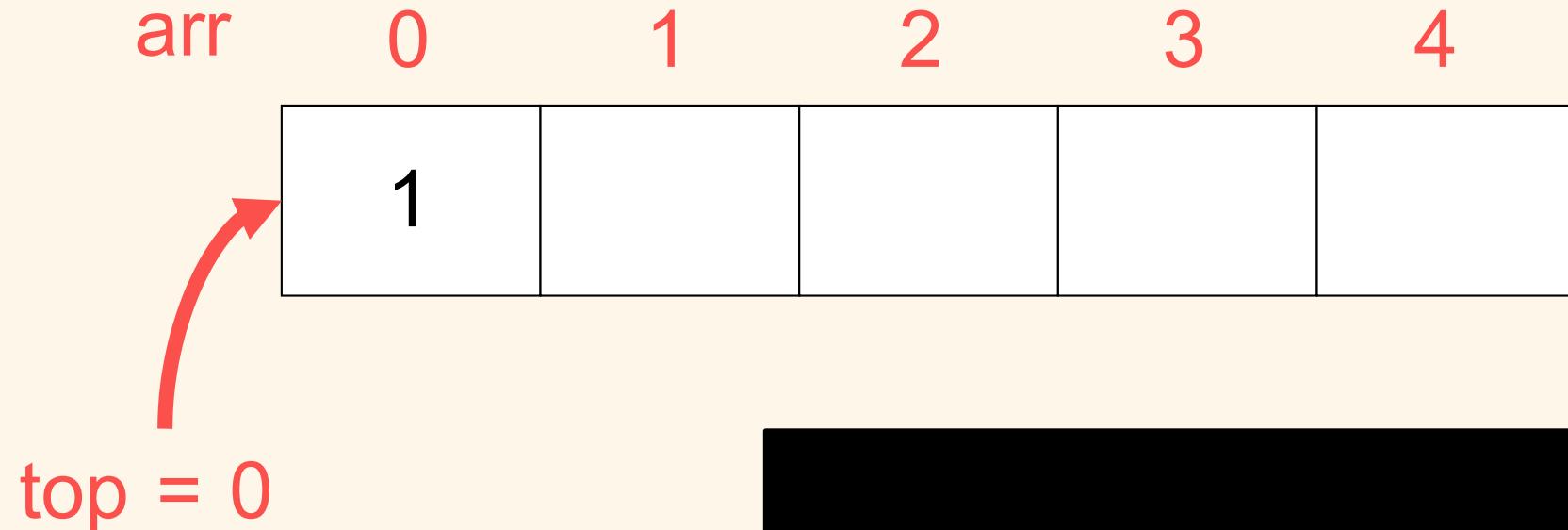
# Array-based Stack



arr[top]



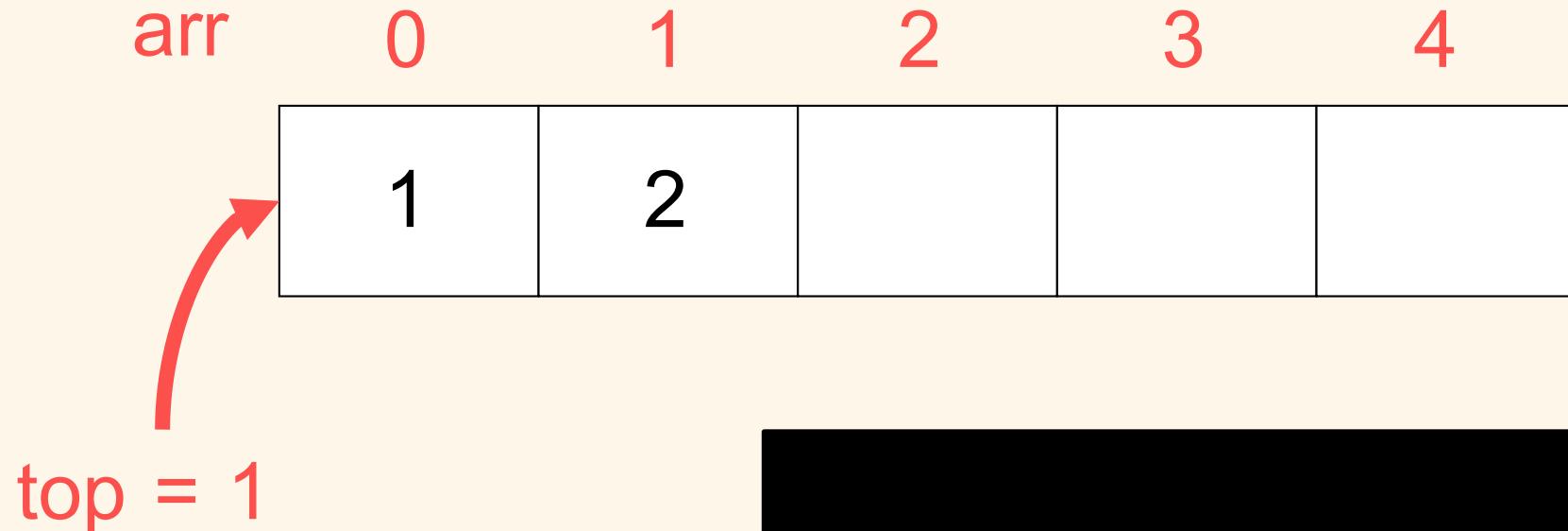
# Array-based Stack



arr[top]



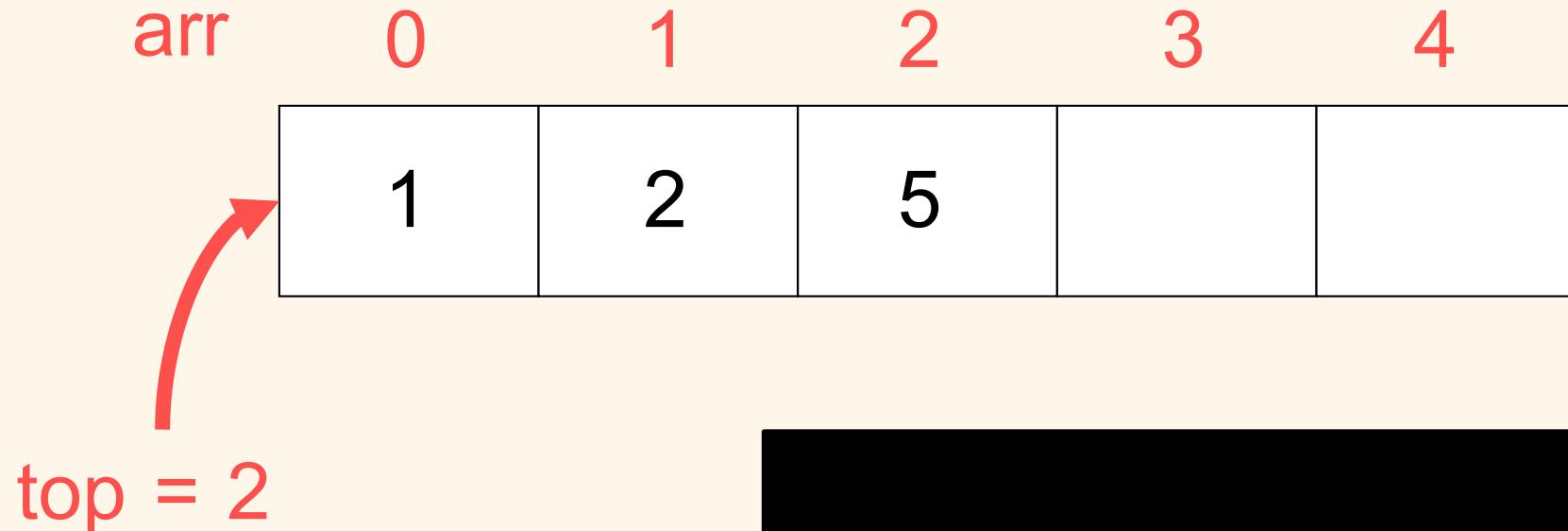
# Array-based Stack



arr[top]



# Array-based Stack



arr[top]



# Array-based Stack

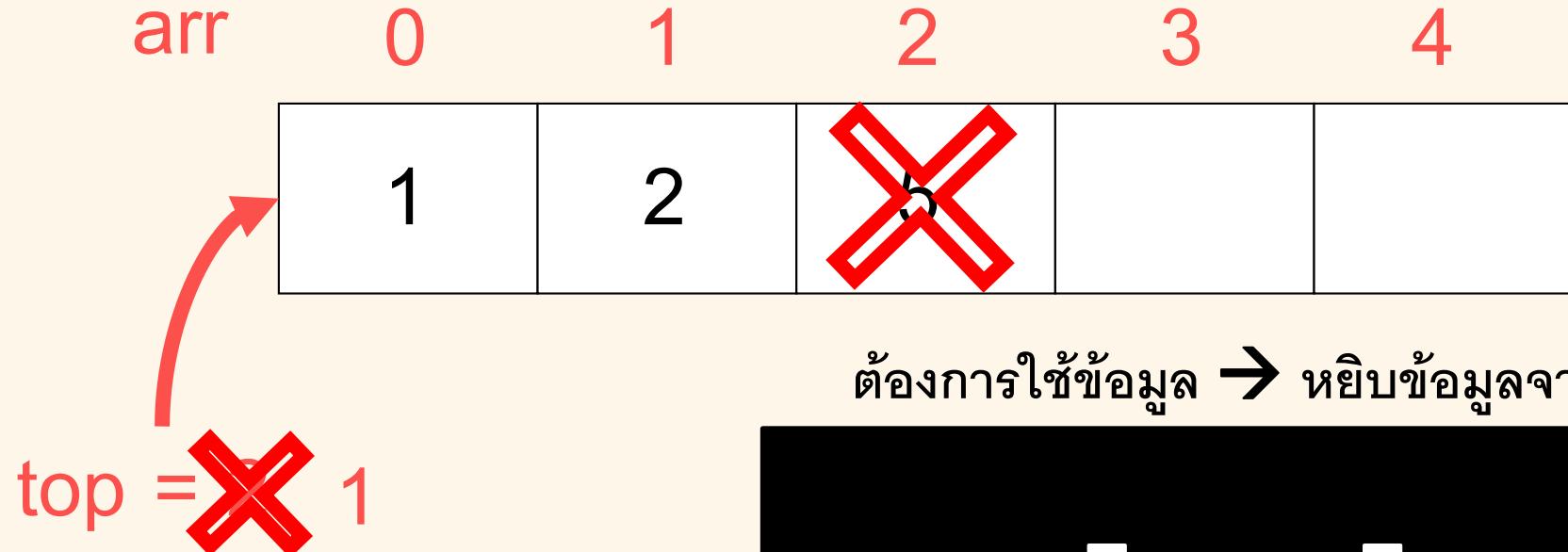


ต้องการใช้ข้อมูล  $\rightarrow$  หยิบข้อมูลจาก index ตัวล่าสุด

arr[top]  $\rightarrow$  5

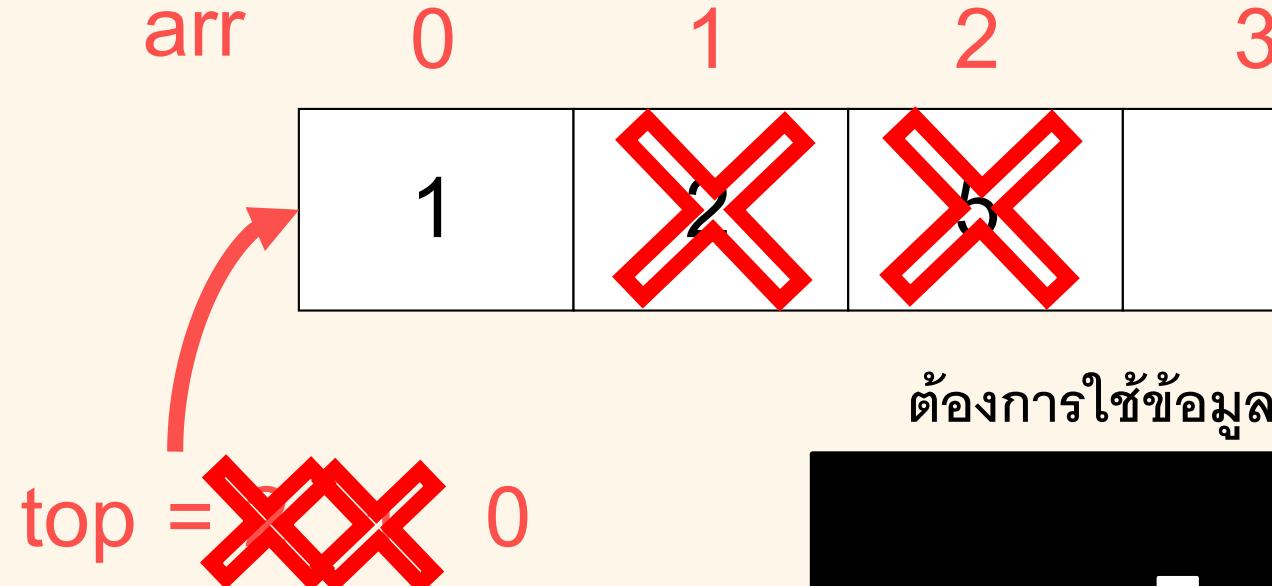


# Array-based Stack





# Array-based Stack



PUSH – นำข้อมูลเข้า

POP – นำข้อมูลออก

ต้องการใช้ข้อมูล →

PEEK, TOP

หยิบข้อมูลตัวล่าสุดของ stack ออกมานะ

arr[top] → 5



# Array-based Stack

PUSH ข้อมูลเกิน array  
= Stack Overflow

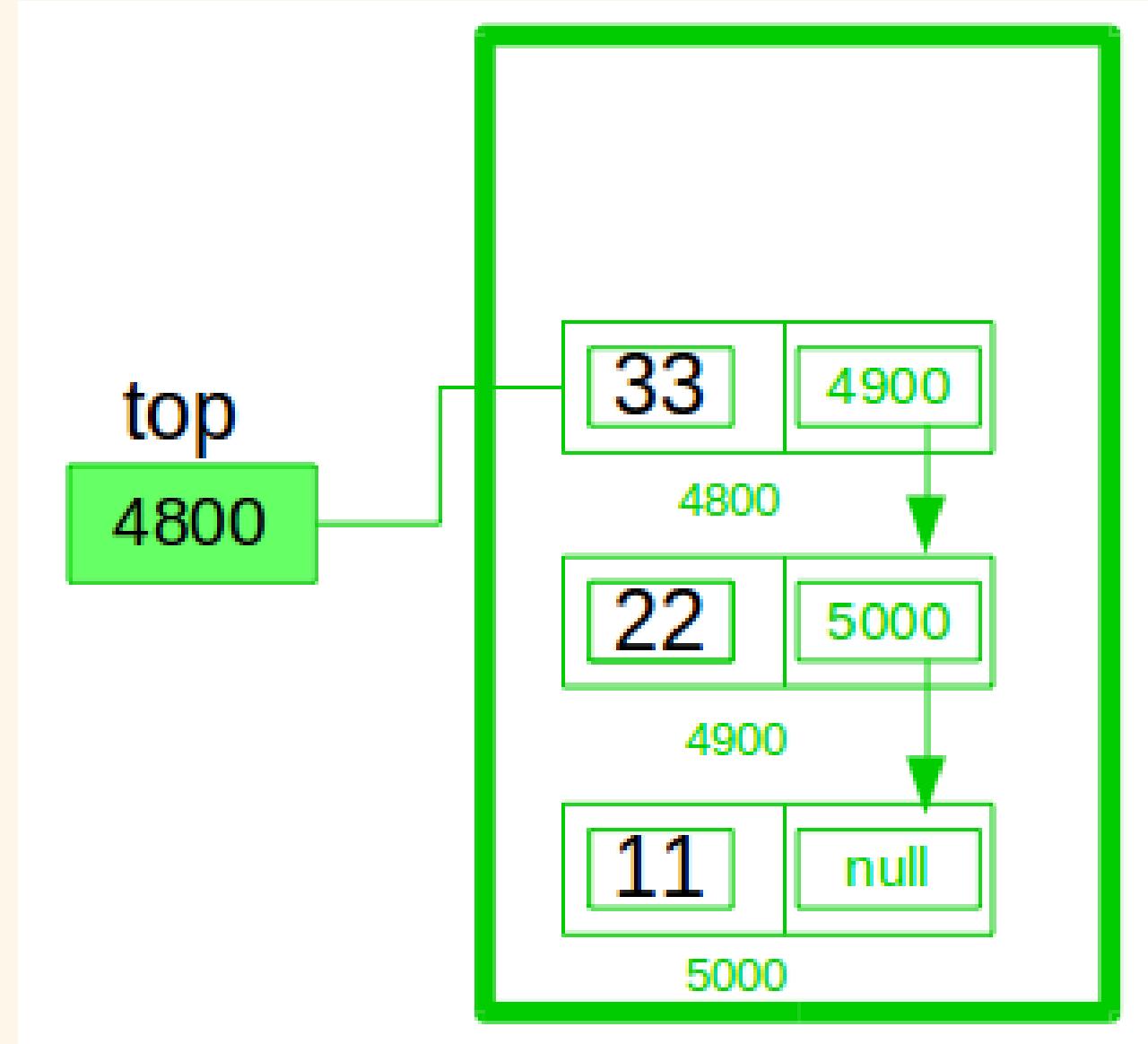


ต้องการใช้ข้อมูล → หยิบข้อมูลจาก index ตัวล่าสุด

arr[top] → 5

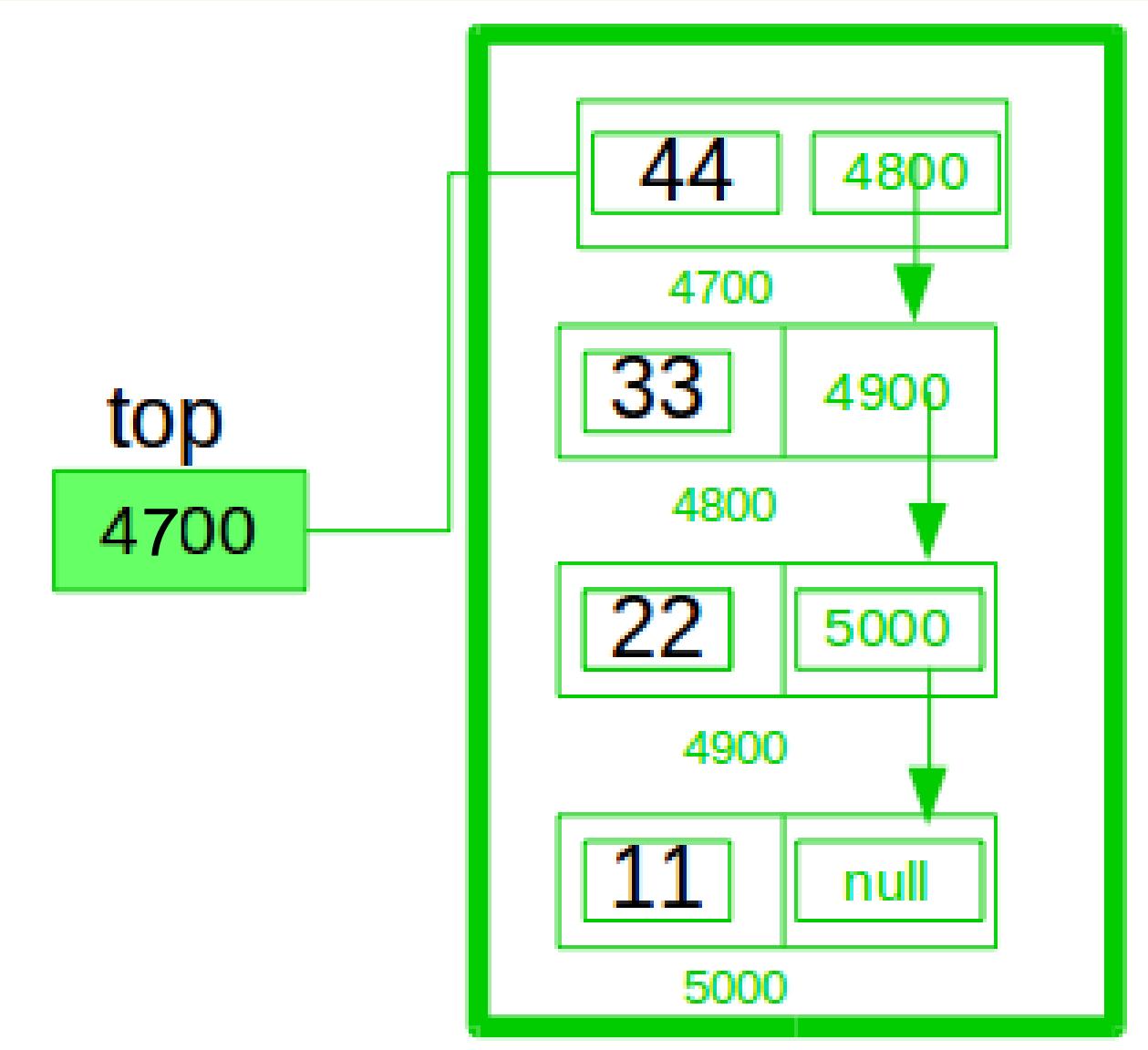


# Linked list based Stack



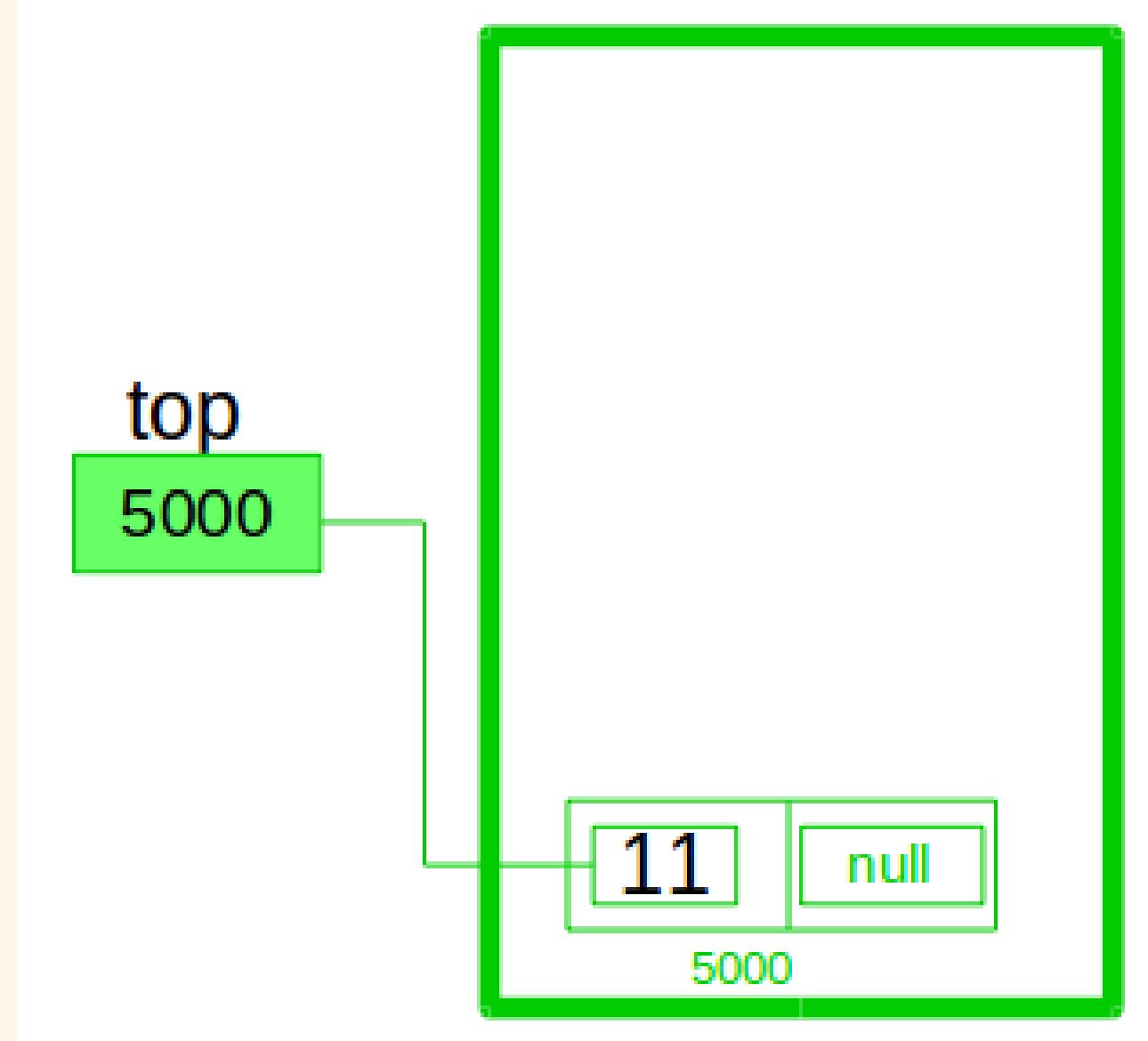


# Linked list based Stack



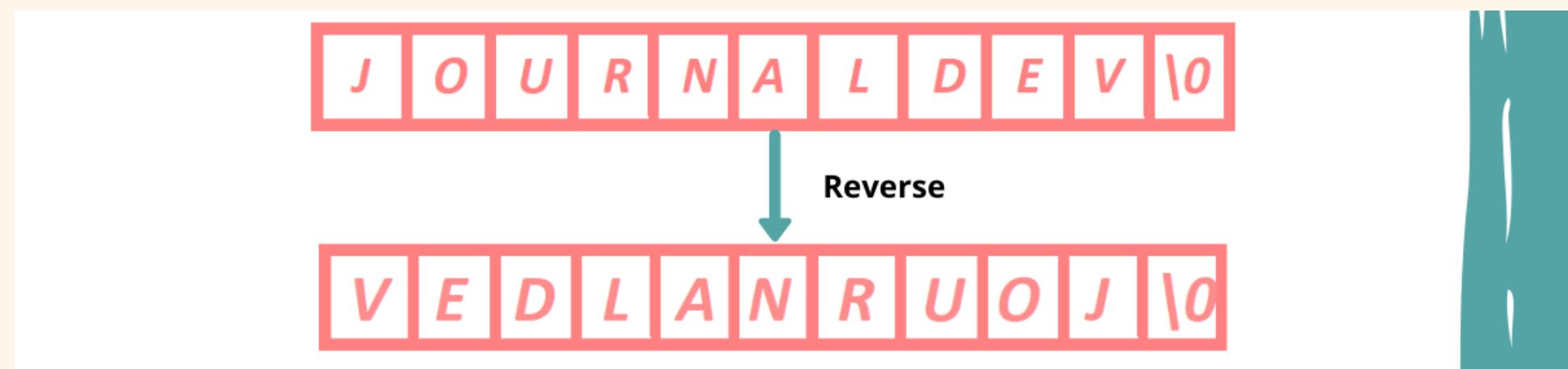
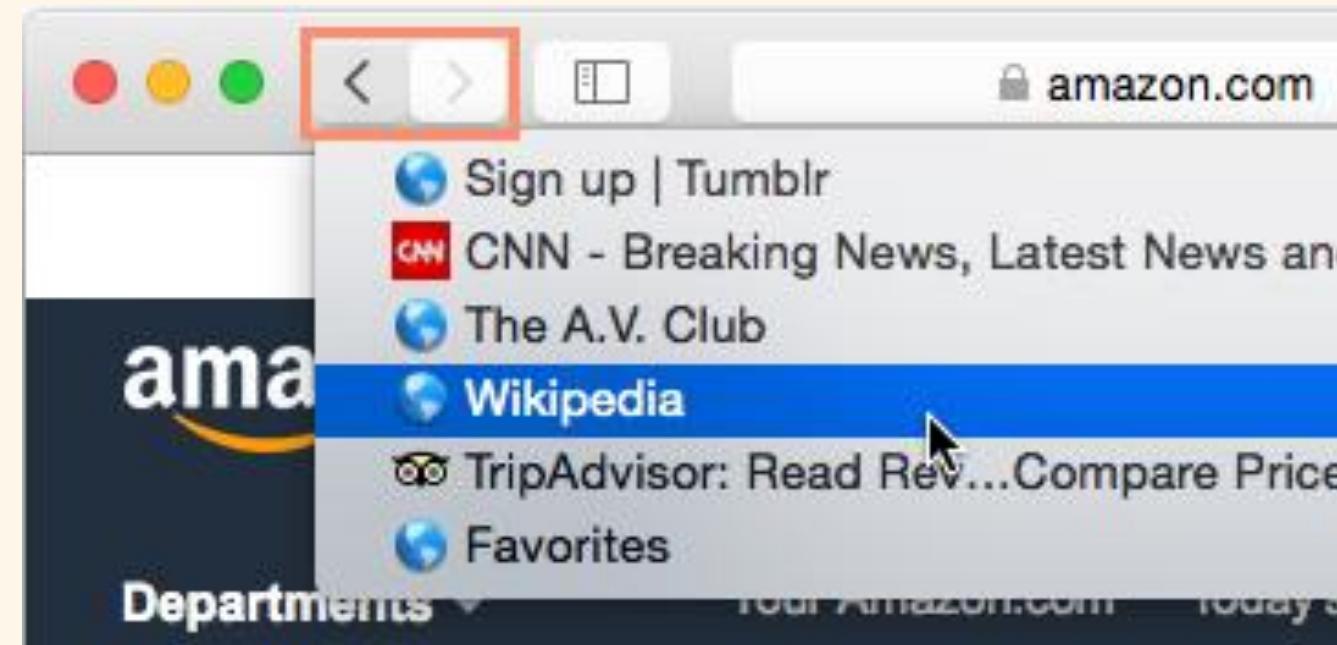
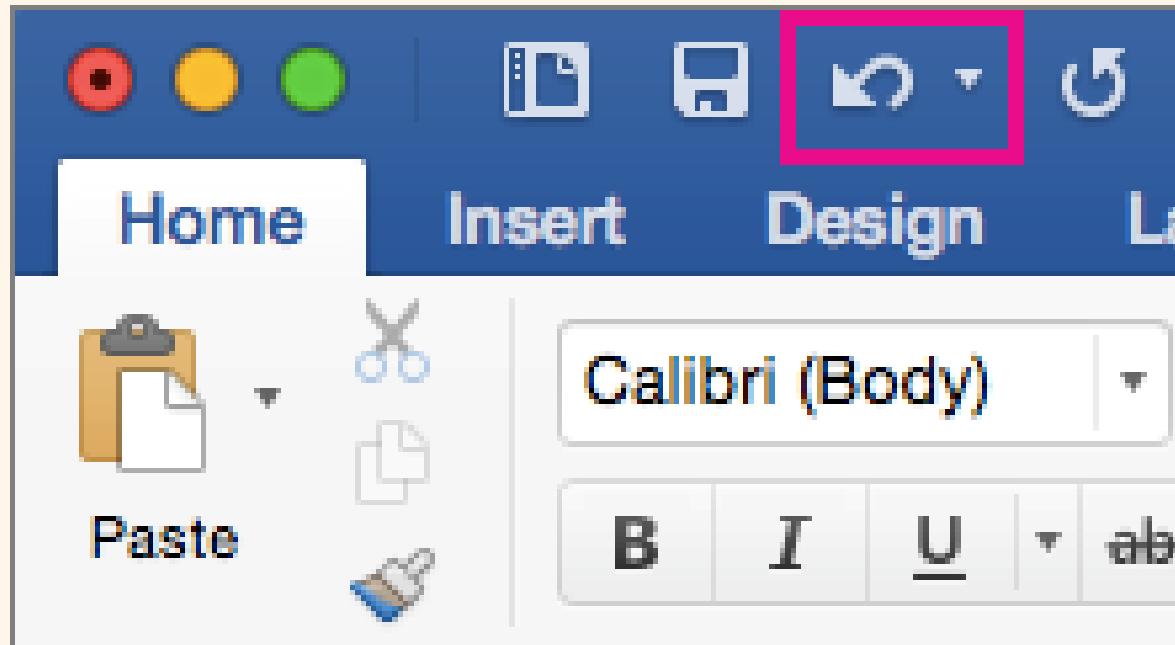


# Linked list based Stack





# สามารถเจอ Stack ได้จากไหน?





# Stack Library

## std::stack

- `push(data)` เพิ่มข้อมูล 'data' เข้าไปที่ด้านบนของ stack
- `pop()` ลบข้อมูลที่อยู่ด้านบนของ stack
- `top()` คืนค่าตัวชี้ของข้อมูลที่อยู่บนสุดของ stack
- `empty()` ตรวจสอบว่า stack ว่างอยู่หรือไม่
- `size()` ตรวจสอบจำนวนข้อมูลที่อยู่ใน stack



## การเลือกใช้โครงสร้างข้อมูล

- การพัฒนาโปรแกรมเพื่อให้ได้โปรแกรมที่มีประสิทธิภาพในการทำงานสูงสุด ต้องคำนึงถึงโครงสร้างข้อมูลที่ใช้
- กรณีที่โปรแกรมไม่ยุ่งยากซับซ้อนมาก ผู้เขียนโปรแกรมไม่จำเป็นต้องคำนึงถึงโครงสร้างข้อมูลมากนัก เพราะโครงสร้างข้อมูลอาจไม่มีผลกับประสิทธิภาพในการทำงานของโปรแกรม
- ในระบบงานใหญ่ที่มีความ слับซับซ้อนมากๆ ต้องคำนึงถึงโครงสร้างข้อมูลที่เลือกใช้ด้วย

ที่มา : อ.ธารารัตน์ พวงศุวรรณ มหาวิทยาลัยบูรพา เรื่อง Data Structures and Algorithms



# Structures Unions Enumerations

In C++

เอกสารประกอบการอบรม

สوان. สาขาวิชาคอมพิวเตอร์ ศูนย์โรงเรียนสวนกุหลาบวิทยาลัย นนทบุรี



# Structures



# Structures

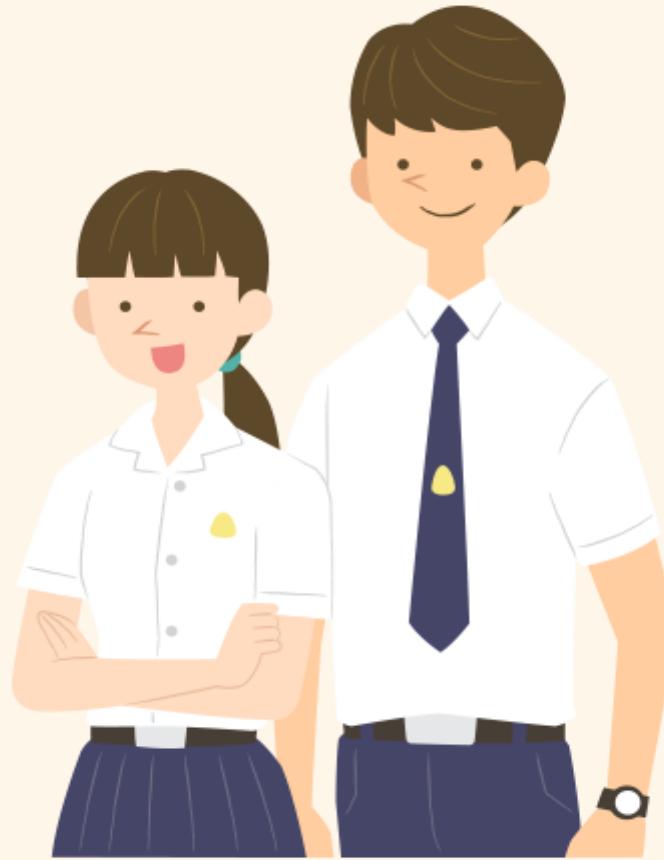


บุคคล

- ชื่อ
- เบอร์โทรศัพท์
- อีเมล



# Structures



## นักศึกษา

- รหัสนักศึกษา
- ชื่อนักศึกษา
- เกรด



# Structures



วันที่

- วัน
- เดือน
- ปี



# Structures



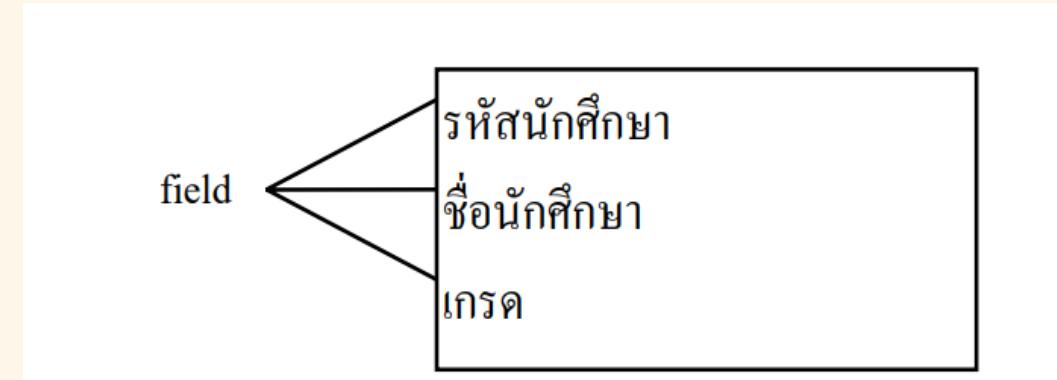
เวลา

- ชั่วโมง
- นาที
- วินาที



# Structures

- Structure คือกลุ่มของตัวประ relay ประเภทรวมตัวกันเป็นประเภทข้อมูลใหม่ประเภทหนึ่ง
- ในฐานข้อมูล เราจะเรียกข้อมูลที่เก็บไว้ว่า ระเบียน (record) ซึ่งประกอบด้วยหลาย ๆ พิล์ด (field) ตัวอย่างเช่น record ของนักศึกษา
- Structure ถูกใช้เพื่อรวมประเภทข้อมูลที่แตกต่างกัน เช่นเดียวกับที่ อาร์เรย์ใช้ในการรวมประเภทข้อมูลที่เหมือนกัน
- Structure จะถูกประกาศโดยใช้ “struct”





# Structures

เริ่มจากปัญหา....

การเขียนโปรแกรมเปรียบเทียบข้อมูลเวลาสองชุด และหาผลต่างเป็นวินาที

**input :** ข้อมูลเวลาสองชุด (แต่ละชุดประกอบด้วย ชั่วโมง นาที วินาที)

**output :** ผลต่างเป็นวินาที



## ແບບໄມ້ໃຊ້

### struct

```
#include <iostream>
#include <cmath>

int main() {
    int h1, h2, m1, m2, s1, s2;

    std::cout << "Time 1\n";
    std::cout << " hour: ";
    std::cin >> h1;
    std::cout << " minute: ";
    std::cin >> m1;
    std::cout << " second: ";
    std::cin >> s1;

    std::cout << "Time 2\n";
    std::cout << " hour: ";
    std::cin >> h2;
    std::cout << " minute: ";
    std::cin >> m2;
    std::cout << " second: ";
    std::cin >> s2;

    int seconds1 = h1 * 3600 + m1 * 60 + s1;
    int seconds2 = h2 * 3600 + m2 * 60 + s2;
    int diff = std::abs(seconds1 - seconds2);

    std::cout << "Seconds difference = " << diff <<
    std::endl;

    return 0;
}
```

## ແບບໃຊ້

### struct

```
#include <iostream>
#include <cmath>

struct Time {
    int h, m, s;
};

void readTime(Time& t) {
    std::cout << " hour: ";
    std::cin >> t.h;
    std::cout << " minute: ";
    std::cin >> t.m;
    std::cout << " second: ";
    std::cin >> t.s;
}

int timeDiff(const Time& t1, const Time& t2) {
    int seconds1 = t1.h * 3600 + t1.m * 60 + t1.s;
    int seconds2 = t2.h * 3600 + t2.m * 60 + t2.s;
    return std::abs(seconds1 - seconds2);
}

int main() {
    Time t1, t2;

    std::cout << "Time 1\n";
    readTime(t1);
    std::cout << "Time 2\n";
    readTime(t2);

    int diff = timeDiff(t1, t2);

    std::cout << "Seconds difference = " << diff << std::endl;

    return 0;
}
```



## แบบไม่ใช้ struct

```
#include <iostream>
#include <cmath>

int main() {
    int h1, h2, m1, m2, s1, s2;

    std::cout << "Time 1\n";
    std::cout << " hour: ";
    std::cin >> h1;
    std::cout << " minute: ";
    std::cin >> m1;
    std::cout << " second: ";
    std::cin >> s1;

    std::cout << "Time 2\n";
    std::cout << " hour: ";
    std::cin >> h2;
    std::cout << " minute: ";
    std::cin >> m2;
    std::cout << " second: ";
    std::cin >> s2;

    int seconds1 = h1 * 3600 + m1 * 60 + s1;
    int seconds2 = h2 * 3600 + m2 * 60 + s2;
    int diff = std::abs(seconds1 - seconds2);

    std::cout << "Seconds difference = " << diff <<
    std::endl;

    return 0;
}
```

## แบบไม่ใช้ struct:

- ตัวแปร h1, m1, s1 สำหรับเวลา 1 และ h2, m2, s2 สำหรับเวลา 2 ถูกประกาศแยกกัน ซึ่งอาจทำให้ยุ่งยาก เมื่อจัดการกับค่าของเวลาหลาย ๆ ชุด
- การคำนวณทำโดยการเข้าถึงตัวแปรแต่ละตัวโดยตรง

```
#include <iostream>
#include <cmath>

int seconds1 = t1.h * 3600 + t1.m * 60 + t1.s;
int seconds2 = t2.h * 3600 + t2.m * 60 + t2.s;
return std::abs(seconds1 - seconds2);

int main() {
    Time t1, t2;

    std::cout << "Time 1\n";
    readTime(t1);
    std::cout << "Time 2\n";
    readTime(t2);

    int diff = timeDiff(t1, t2);

    std::cout << "Seconds difference = " << diff << std::endl;

    return 0;
}
```



## แบบใช้ struct:

- struct Time จะรวมชั่วโมง, นาที และวินาทีเข้าเป็นหน่วยเดียว ซึ่งทำให้โค้ดสะอาดขึ้น โดยเฉพาะเมื่อต้องจัดการกับค่าของเวลาหลาย ๆ ชุด
- ฟังก์ชัน readTime และ timeDiff ทำงานกับวัตถุประเภท Time ทำให้โค้ดเป็นแบบโมดูลาร์มากขึ้นและง่ายต่อการดูแล
- การใช้ struct ทำให้โค้ดมีระเบียบและเป็นระบบมากขึ้น ทำให้ง่ายต่อการจัดการเมื่อโค้ดขยายตัวหรือเมื่อจัดการกับสถานการณ์ที่ซับซ้อนมากขึ้น

```
std::endl;
```

```
    return 0;  
}
```

```
#include <iostream>  
#include <cmath>  
  
struct Time {  
    int h, m, s;  
};  
  
void readTime(Time& t) {  
    std::cout << " hour: ";  
    std::cin >> t.h;  
    std::cout << " minute: ";  
    std::cin >> t.m;  
    std::cout << " second: ";  
    std::cin >> t.s;  
}  
  
int timeDiff(const Time& t1, const Time& t2) {  
    int seconds1 = t1.h * 3600 + t1.m * 60 + t1.s;  
    int seconds2 = t2.h * 3600 + t2.m * 60 + t2.s;  
    return std::abs(seconds1 - seconds2);  
}  
  
int main() {  
    Time t1, t2;  
  
    std::cout << "Time 1\n";  
    readTime(t1);  
    std::cout << "Time 2\n";  
    readTime(t2);  
  
    int diff = timeDiff(t1, t2);  
  
    std::cout << "Seconds difference = " << diff << std::endl;  
  
    return 0;  
}
```



## ແບບໄມ້ໃຊ້ struct

```
#include <iostream>
#include <cmath>

int main() {
    int h1, h2, m1, m2, s1, s2;

    std::cout << "Time 1\n";
    std::cout << " hour: ";
    std::cin >> h1;
    std::cout << " minute: ";
    std::cin >> m1;
    std::cout << " second: ";
    std::cin >> s1;

    std::cout << "Time 2\n";
    std::cout << " hour: ";
    std::cin >> h2;
    std::cout << " minute: ";
    std::cin >> m2;
    std::cout << " second: ";
    std::cin >> s2;

    int seconds1 = h1 * 3600 + m1 * 60 + s1;
    int seconds2 = h2 * 3600 + m2 * 60 + s2;
```

```
}  
return 0;
```

## ແບບໃຊ້ struct

```
#include <iostream>
#include <cmath>

struct Time {
    int h, m, s;
};

void readTime(Time& t) {
    std::cout << " hour: ";
    std::cin >> t.h;
    std::cout << " minute: ";
    std::cin >> t.m;
    std::cout << " second: ";
    std::cin >> t.s;
}

int timeDiff(const Time& t1, const Time& t2) {
    int seconds1 = t1.h * 3600 + t1.m * 60 + t1.s;
    int seconds2 = t2.h * 3600 + t2.m * 60 + t2.s;
    return std::abs(seconds1 - seconds2);
}

int main() {
    Time t1, t2;

    std::cout << "Time 1\n";
    readTime(t1);
    std::cout << "Time 2\n";
}
```

```
}  
return 0;
```

<https://github.com/hchaser/posn66>



## ແບບໄມ້ໃຊ້ struct

```
#include <iostream>
#include <cmath>

int main() {
    int h1, h2, m1, m2, s1, s2;

    std::cout << "Time 1\n";
    std::cout << " hour: ";
    std::cin >> h1;
    std::cout << " minute: ";
    std::cin >> m1;
    std::cout << " second: ";
    std::cin >> s1;

    std::cout << "Time 2\n";
    std::cout << " hour: ";
    std::cin >> h2;
    std::cout << " minute: ";
    std::cin >> m2;
    std::cout << " second: ";
    std::cin >> s2;

    int seconds1 = h1 * 3600 + m1 * 60 + s1;
    int seconds2 = h2 * 3600 + m2 * 60 + s2;
```

```
}
```

## ແບບໃຊ້ struct

```
#include <iostream>
#include <cmath>

struct Time {
    int h, m, s;
};

void readTime(Time& t) {
    std::cout << " hour: ";
    std::cin >> t.h;
    std::cout << " minute: ";
    std::cin >> t.m;
    std::cout << " second: ";
    std::cin >> t.s;
}

int timeDiff(const Time& t1, const Time& t2) {
    int seconds1 = t1.h * 3600 + t1.m * 60 + t1.s;
    int seconds2 = t2.h * 3600 + t2.m * 60 + t2.s;
    return std::abs(seconds1 - seconds2);
}

int main() {
    Time t1, t2;

    std::cout << "Time 1\n";
    readTime(t1);
    std::cout << "Time 2\n";
```

```
}
```

<https://github.com/hchaser/posn66>



# Structures

Struct keyword

tag or structure tag

```
struct geeksforgeeks  
{  
    char _name [10];  
    int id [5];  
    float salary;  
};
```

Members or  
Fields of structure

struct ชื่อสตรัคเจอร์

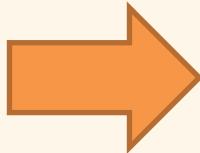
{

ชนิดข้อมูล ชื่อรายการ;  
ชนิดข้อมูล ชื่อรายการ;  
ชนิดข้อมูล ชื่อรายการ;  
ชนิดข้อมูล ชื่อรายการ;



# Syntax of Struct

```
struct structureName {  
    dataType member1;  
    dataType member2;  
    ...  
};
```



```
struct Person {  
    char name[50];  
    int id[5];  
    float salary;  
};
```



# Structures

```
#include <iostream>
#include <string>
using namespace std;

struct Student {
    int ID;          // เก็บ ID นักเรียน
    string name;    // เก็บชื่อของนักเรียน
    float score;    // เก็บคะแนนของนักเรียน
};
```

Student ID : 12345  
Student Name :  
Kongkiat Jedipharm  
Student Score : 88.5

```
int main()
{
    // ประกาศตัวแปร struct นักเรียน
    Student student1;

    // กำหนดค่าข้อมูลให้กับตัวแปร student1
    student1.ID = 12345;
    student1.name = "Kongkiat Jedipharm";
    student1.score = 88.5;

    // แสดงข้อมูลของนักเรียน
    cout << "Student ID : " << student1.ID << endl;
    cout << "Student Name : " << student1.name << endl;
    cout << "Student Score : " << student1.score << endl;

    return 0;
}
```



# typedef เป็นคำสั่งในภาษา C++ ที่ใช้สำหรับการกำหนดชื่อใหม่ให้กับประเภทข้อมูลที่มือyu'แล้ว

ตัวอย่างโปรแกรม C++ ที่แสดงการใช้ struct ร่วมกับ typedef:

```
#include <iostream>
#include <string>
using namespace std;

typedef struct Student {
    int ID;          // เก็บ ID นักเรียน
    string name;     // เก็บชื่อของนักเรียน
    float score;     // เก็บคะแนนของนักเรียน
} StudentInfo;
```

```
int main()
{
    // สร้างตัวแปรนักเรียนโดยใช้typedef
    StudentInfo student1;

    // กำหนดค่าข้อมูลให้กับนักเรียน
    student1.ID = 12345;
    student1.name = "Kongkiat Jedipharm";
    student1.score = 88.5;

    // แสดงข้อมูลของนักเรียน
    cout << "Student ID : " << student1.ID << endl;
    cout << "Student Name : " << student1.name << endl;
    cout << "Student Score : " << student1.score << endl;

    return 0;
}
```



# typedef เป็นคำสั่งในภาษา C++ ที่ใช้สำหรับการกำหนดชื่อใหม่ให้กับประเภทข้อมูลที่มีอยู่แล้ว

ตัวอย่างโปรแกรม C++ ที่แสดงการใช้ struct ร่วมกับ typedef:

```
#include <iostream>
#include <string>
using namespace std;

typedef struct Student {
    int ID;          // เก็บ ID นักเรียน
    string name;     // เก็บชื่อของนักเรียน
    float score;     // เก็บคะแนนของนักเรียน
} StudentInfo;
```

```
int main()
{
    // สร้างตัวแปรนักเรียนโดยใช้ typedef
    StudentInfo student1;

    // กำหนดค่าข้อมูลให้กับนักเรียน
    student1.ID = 12345;
    student1.name = "Kongkiat Jedipharm";
    student1.score = 88.5;

}
```

typedef ถูกใช้เพื่อกำหนดชื่อใหม่ให้กับโครงสร้าง struct ของนักเรียน ซึ่งในกรณีนี้เราตั้งชื่อว่า StudentInfo ดังนั้นเราสามารถใช้ StudentInfo แทน struct Student ได้ในโปรแกรม



ผู้พัฒนาโปรแกรมสามารถสร้างคำเหนื่อน (aliases) เพื่อกำหนดประเภทข้อมูลต่าง ๆ ทั้งที่มีอยู่แล้วและกำหนดขึ้นมาเองใหม่ได้โดยใช้ `typedef`  
ตัวอย่างเช่น

```
1. typedef int date;
2. typedef struct {
    char *face;
    char *suit;
} Card;
```

เมื่อกำหนดประเภทข้อมูลใหม่แล้วเราก็สามารถเรียกใช้ประเภทข้อมูลนี้ได้ดังนี้  
1. `date day, month, year;`  
2. `Card card;`



# Unions



- Union เป็นประเภทของ structure ที่สามารถใช้ได้เมื่อปริมาณหน่วยความจำที่ใช้เป็นปัจจัยสำคัญ
- คล้ายกับ Structure โดย Union สามารถมีประเภทข้อมูลหลายประเภทได้ แต่ต่างจาก struct ตรงที่ทุกครั้งที่มีการกำหนดค่าใหม่ให้กับตัวแปรใน Union จะไปเขียนทับค่าก่อนหน้า
- Union มีประโยชน์มากในกรณีที่ไม่ทราบประเภทข้อมูลที่ถูกส่งผ่านฟังก์ชัน การใช้ Union ที่ประกอบไปด้วยประเภทข้อมูลทุกแบบที่เป็นไปได้สามารถแก้ปัญหานี้ได้
- Union ถูกประกาศโดยใช้คีย์เวิร์ด "union"

รูปแบบ union <ชื่อประเภทข้อมูล> {  
    <ประเภทข้อมูลของสมาชิกตัวที่1>   <ชื่อตัวแปรของสมาชิกตัวที่1>;  
    <ประเภทข้อมูลของสมาชิกตัวที่2>   <ชื่อตัวแปรของสมาชิกตัวที่2>;  
    .  
    .  
    .  
    <ประเภทข้อมูลของสมาชิกตัวที่n>   <ชื่อตัวแปรของสมาชิกตัวที่n>;  
};

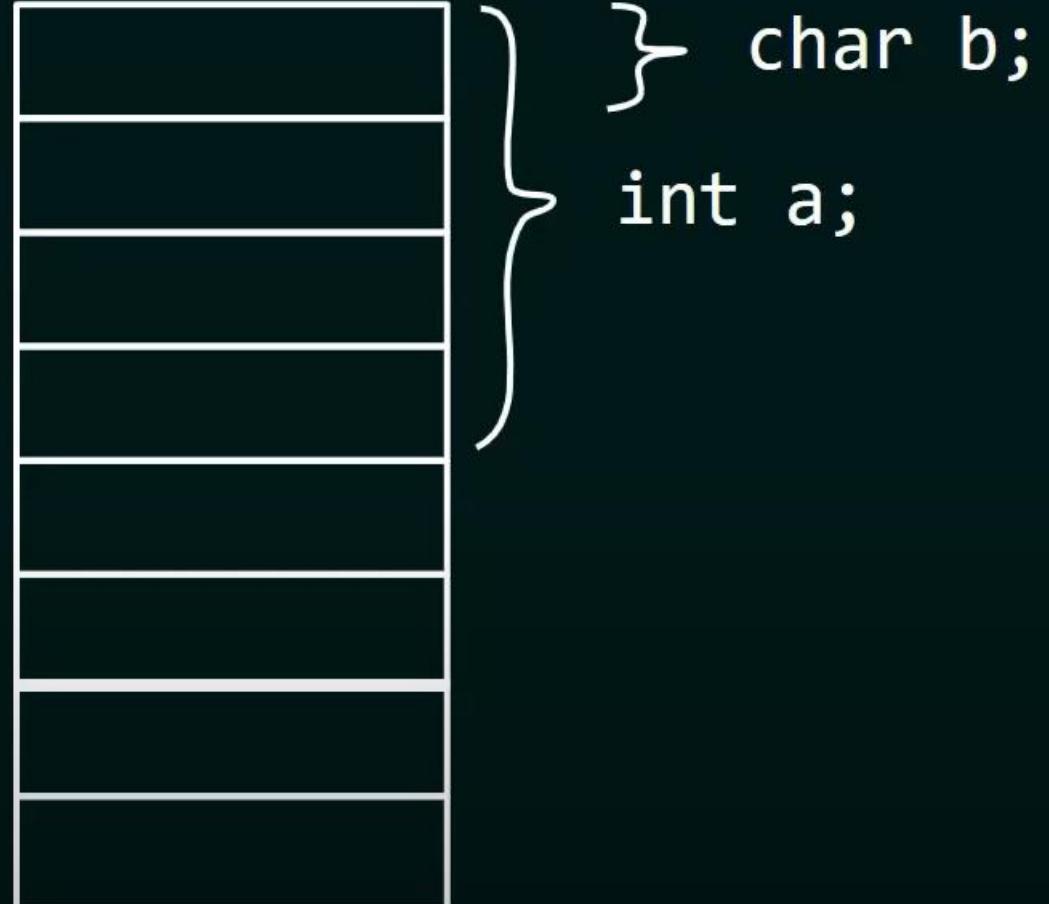
ตัวอย่างเช่น union number{  
    int x;  
    double y;  
};



```
struct abc
```



```
union abc
```





```
#include <iostream>
using namespace std;

// การกำหนด Union
union Data {
    int intValue;
    char charValue;
    float floatValue;
};
```

```
// พิจารณาหลัก
int main()
{
    // การกำหนดค่าให้กับ Union
    union Data data;

    data.intValue = 34;

    // แสดงผลค่าที่ถูกจัดเก็บในหน่วยความจำ
    cout << "The first value at the allocated memory: " << data.intValue << endl;

    data.charValue = 'A';

    cout << "The next value stored after removing the previous value: " << data.charValue << endl;

    data.floatValue = 34.34;

    cout << "The Final value value at the same allocated memory space: " << data.floatValue << endl;

    return 0;
}
```



```
#include <iostream>
using namespace std;
```

### // การกำหนด Union

```
union Data {
    int intValue;
    char charValue;
    float floatValue;
```

```
// พิมพ์ชั้นหลัก
int main()
{
    // การกำหนดค่าให้กับ Union
    union Data data;

    data.intValue = 34;

    // แสดงผลค่าที่ถูกจัดเก็บในหน่วยความจำ
    cout << "The first value at the allocated memory: " << data.intValue << endl;
```

เมื่อเรากำหนดค่า 'A' ให้กับตัวแปร **charValue** ซึ่งเป็นประเภท **char** ค่าเก่าที่ถูกเก็บไว้ใน **inValue** (ค่า 34) จะถูกลบออกและแทนที่ด้วยค่าใหม่ที่เป็นตัวอักษร 'A' ซึ่งถูกเก็บในพื้นที่หน่วยความจำเดียวกันกับ **inValue** และสามารถแสดงผลอย่างถูกต้องใน **Output**

The first value at the allocated memory : 34

The next value stored after removing the previous value : A

The Final value value at the same allocated memory space : 34.34

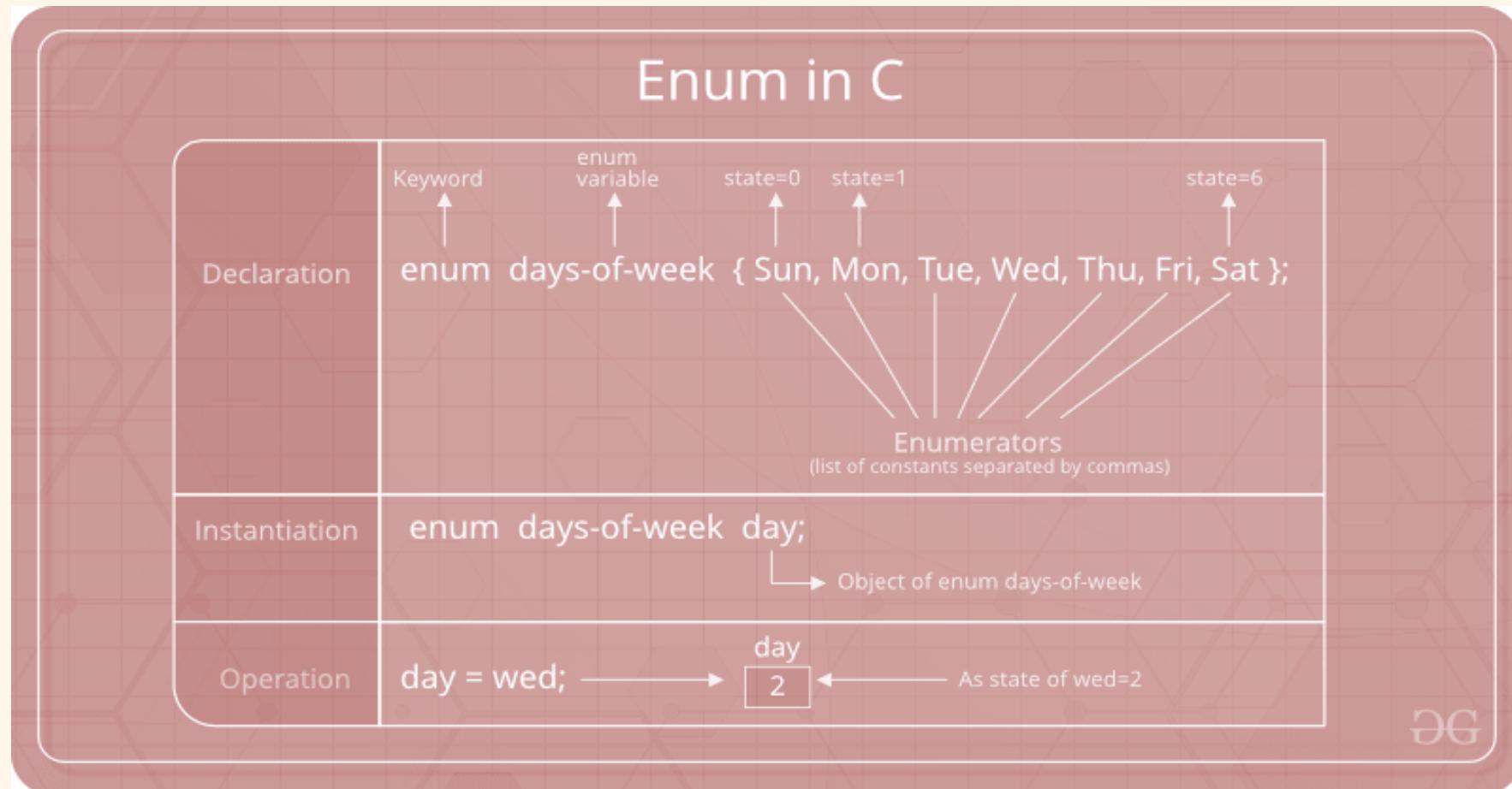


# Enumeration



# Enumeration

- Enums คือชนิดข้อมูลที่กำหนดขึ้นเองโดยผู้ใช้ ซึ่งประกอบไปด้วยค่าคงที่แบบจำนวนเต็มที่มีชื่อกำกับ
- Enums ช่วยในการกำหนดค่าคงที่ให้กับชุดของชื่อ เพื่อทำให้โปรแกรมอ่านง่ายขึ้น, ดูแลรักษาง่ายขึ้น, และเข้าใจได้ง่ายขึ้น
- การประกาศ Enums ใช้คีย์เวิร์ด "enum"





# Enumeration

- Enums คือชนิดข้อมูลที่กำหนดขึ้นเองโดยผู้ใช้ ซึ่งประกอบไปด้วยค่าคงที่แบบจำนวนเต็มที่มีชื่อกำกับ
- Enums ช่วยในการกำหนดค่าคงที่ให้กับชุดของชื่อ เพื่อทำให้โปรแกรมอ่านง่ายขึ้น, ดูแลรักษาง่ายขึ้น, และเข้าใจได้ง่ายขึ้น
- การประกาศ Enums ใช้คีย์เวิร์ด "enum"

```
#include <iostream>
using namespace std;

enum Status { Status1, Status2,
Status3 };

Status S1 = Status1;
Status S2 = Status2;
Status S3 = Status3;
```

```
int main()
{
    cout << "The numerical value "
        << "assigned to Status1 : " << S1 << endl;

    cout << "The numerical value "
        << "assigned to Status2 : " << S2 << endl;

    cout << "The numerical value "
        << "assigned to Status3 : " << S3 << endl;

    return 0;
}
```



# Enumeration

- Enums คือชนิดข้อมูลที่กำหนดขึ้นเองโดยผู้ใช้ ซึ่งประกอบไปด้วยค่าคงที่แบบจำนวนเต็มที่มีชื่อกำกับ
- Enums ช่วยในการกำหนดค่าคงที่ให้กับชุดของชื่อ เพื่อทำให้โปรแกรมอ่านง่ายขึ้น, ดูแลรักษาง่ายขึ้น, และเข้าใจได้ง่ายขึ้น
- การประกาศ Enums ใช้คีย์เวิร์ด "enum"

```
#include <iostream>
using namespace std;

enum Status { Status1, Status2,
Status3 };

Status S1 = Status1;
```

```
int main()
{
    cout << "The numerical value "
        << "assigned to Status1 : " << S1 << endl;

    cout << "The numerical value "
        << "assigned to Status2 : " << S2 << endl;

    cout << "The numerical value "
        << "assigned to Status3 : " << S3 << endl;
```

The numerical value assigned to Status1 : 0

The numerical value assigned to Status2 : 1

The numerical value assigned to Status3 : 2



## ทดสอบเขียน enumeration

```
#include <iostream>

enum year { Jan = 1, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec };

int main() {
    for (int i = Jan; i <= Dec; i++) {
        std::cout << i << " ";
    }

    return 0;
}
```

```
C:\Users\Ohayou\OneDrive\Desktop\pointer\enum2.exe
1 2 3 4 5 6 7 8 9 10 11 12
-----
Process exited after 0.04197 seconds with return value 0
Press any key to continue . . .
```



# Nested Structures (โครงสร้างชั้นในโครงสร้าง)

เมื่อได้ที่ structure หนึ่งอยู่ใน structure หนึ่งซึ่งกลายเป็นโครงสร้างข้อมูลใหม่ เราเรียกว่า nested structure

ตัวอย่างเช่น

```
enum blood_type{unknown, A, B, AB, O};  
enum rh_factor{negative, positive};  
struct blood_pressure{  
    int systolic;  
    int diastolic;  
};  
struct donor_info{  
    enum blood_type type;  
    enum rh_factor rh;  
    struct blood_pressure bp;  
    int heart_rate;  
};
```

การเข้าถึงสมาชิกของ nested structure ต้องใช้ . (dot operator) 2 ครั้ง เช่น current\_donor.bp.systolic = 130;



# Nested Structures (โครงสร้างชั้นในโครงสร้าง)

```
#include <iostream>

enum blood_type { unknown, A, B, AB, O };
enum rh_factor { negative, positive };

struct blood_pressure {
    int systolic;
    int diastolic;
};

struct donor_info {
    blood_type type;
    rh_factor rh;
    blood_pressure bp;
    int heart_rate;
};

int main() {
    donor_info current_donor;
    current_donor.bp.systolic = 130;
    std::cout << "systolic : " << current_donor.bp.systolic << std::endl;
    return 0;
}
```



# Nested Structures (โครงสร้างชั้นในโครงสร้าง)

แบบฝึกหัด จาก Struct ที่กำหนดให้ ให้  
นักเรียนรับค่าดังนี้

- ชื่อ
  - อายุ
  - เมือง
  - ถนน
- แล้วแสดงผล



# File In C++

เอกสารประกอบการอบรม  
สوان. สาขาวิชาคอมพิวเตอร์ ศูนย์โรงเรียนสวนกุหลาบวิทยาลัย นนทบุรี



```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ofstream myFile("score.txt", ios::out);

    if (!myFile) { // ตรวจสอบว่าเปิดไฟล์สำเร็จหรือไม่
        cout << "Error opening file!" << endl;
        return 1;
    }

    // เขียนข้อมูลไปที่ไฟล์
    myFile << "This is a test score." << endl;

    myFile.close(); // ปิดไฟล์

    return 0;
}
```

สร้างไฟล์ เขียนไฟล์



```
#include <iostream>
#include <fstream>
#include <cstring>

using namespace std;

int main()
{
    ofstream myFile("score.txt", ios::out);
    char name[30];
    int score; // เพิ่มเครื่องหมาย ;

    cout << "Enter name and score (type 'exit' as name to stop): " << endl;
    while (true)
    {
        cin >> name;
        if (strcmp(name, "exit") == 0) // ถ้าพิมพ์ 'exit' จะหยุดการป้อนข้อมูล
            break;

        cin >> score; // อ่านคะแนน
        myFile << name << '\t' << score << '\n'; // เขียนชื่อและคะแนนไปที่ไฟล์
    }

    myFile.close(); // ปิดไฟล์

    cout << "Data saved successfully." << endl;
    return 0;
}
```

รับข้อมูลจากผู้ใช้ และเขียนไฟล์



## อ่านข้อมูลจากไฟล์

```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream readFile("score.txt", ios::in);
    char name[30];
    int score;

    // ตรวจสอบว่าเปิดไฟล์สำเร็จหรือไม่
    if (!readFile) {
        cout << "Error opening file!" << endl;
        return 1; // ออกจากโปรแกรมหากไม่สามารถเปิดไฟล์
    }

    // อ่านข้อมูลจากไฟล์
    while (readFile >> name >> score) {
        cout << name << ' ' << score << endl; // แสดงชื่อและคะแนน
    }

    readFile.close(); // ปิดไฟล์
    return 0;
}
```



## แหล่งอ้างอิง

GeeksforGeeks - Data Structures <https://www.geeksforgeeks.org/data-structures/>

GeeksforGeeks - Linked List in C++ <https://www.geeksforgeeks.org/linked-list-set-1-introduction/>

GeeksforGeeks - Stack in C++ <https://www.geeksforgeeks.org/stack-data-structure-introduction-program/>

GeeksforGeeks - Queue in C++ <https://www.geeksforgeeks.org/queue-cpp-stl/>

เอกสารการสอน อ.ธาราัรัตน์ พวงสุวรรณ มหาวิทยาลัยบูรพา เรื่อง Data Structures and Algorithms

Mikelopster <https://docs.mikelopster.dev/>

<https://preecha11th.wordpress.com/>

cplusplus.com <http://wwwcplusplus.com/>

LearnCPP.com <https://www.learnCPP.com/>

Stack Overflow <https://stackoverflow.com/>

Tutorialspoint <https://www.tutorialspoint.com/cplusplus/index.htm>

Programiz - C++ Programming <https://www.programiz.com/cpp-programming>

Cppreference.com <https://en.cppreference.com/w/>