同步更新於https://github.com/mickey9910326/Homework/tree/master/DataStructure/HW4

合作同學：

102303523謝定翰

102303021吳承翰

**HW4\_Q1**

**The Question**

Give a doubly linked list with length n, which save odd number from 1 to 2n-1  
Ex: n = 3,  
list : 1 ⇔ 3 ⇔ 5  
Now, write a program to add an even number m into this list. Also, put m at the appropriate position of the list in order to keep the sequence of the list.  
Ex: m = 4,  
list after add 4 : 1 ⇔ 3 ⇔ 4 ⇔ 5

**My Answer**

#include <iostream>

using namespace std;

class node {

private:

node \*\_prev;

node \*\_next;

int \_data;

public:

node(){ //建構子

\_prev = NULL;

\_next = NULL;

\_data = 0;

};

node \*prev(){return \_prev;}; //取出 \_prev

node \*next(){return \_next;}; //取出 \_next

int data(){return \_data;}; //取出 \_data

void prev(node\* n){\_prev = n;}; //寫入 \_prev

void next(node\* n){\_next = n;}; //寫入 \_next

void data(int d) {\_data = d;}; //寫入 \_data

void show();

};

class list {

private:

node\* root\_node;

public:

list(int n){ //建構子 依題目要求建構出陣列

/\*\*

\* TODO

\* @var cur\_node 存放當前節點

\* @var new\_node 愈加入之新節點

\* 先令 cur\_node = root\_node

\* cur\_node 不斷往後走(for @var i=0)

\* 每次產生新節點放入new\_node，給值(2\*i+1)

\* new\_node 與 cur\_node 連接

\* 直到(i > n)

\* 或下個節點的直大於n(cur\_node->data() >= n)

\*/

};

node\* root(){return root\_node;};

void show(); //印出當前陣列，code在下部

void add(int n);//依題目要求加入節點，code在下部

};

int main() {

/\*\*

\* TODO

\* 輸入list長度n

\* 判斷n

\* 產生list list(n)

\* 令 cur\_node = root\_node

\* cur\_node不斷往後走(while)

\* 直到下個節點為空(cur\_node->next()==NULL)

\* 或下個節點的直大於n(cur\_node->data() >= n)

\*/

return 0;

}

void list::show(){

/\*\*

\* TODO

\* show all the node in the list

\*/

}

void list::add(int n){

/\*\*

\* TODO

\* 插入數字n到list中

\* @var cur\_node 存放當前節點

\* @var new\_node 愈加入之新節點

\* 令 cur\_node = root\_node

\* cur\_node不斷往後走(while)

\* 直到下個節點為空(cur\_node->next()==NULL)

\* 或下個節點的直大於n(cur\_node->data() >= n)

\*/

}

## HW4\_Q3

##### The Question

Give a recursive algorithm to output all permutations of 1,..., n in lexicographical order.

##### My Answer

// HW4\_Q3

// TODO

// 使用Backtracking實作

// 請看CODE

## HW4\_Q5

##### The Question

How to delete the first node of a singly list ? Assume a list L with length m.

##### My Answer

void delete\_node(int node\_num) { //node\_num = m

/\*\*

\* TODO

\* @var cur\_node 存放當前節點

\* @var node\_num 愈刪除之節點編號(1開頭)

\* cur\_node = head\_node

\* @var i = 1 計數用

\* 當 (i != m-1) 且 (cur\_node->next != NULL)

\* cur\_node 不斷往下個節點走

\* 將cur\_node->next (愈刪除之節點)放入 tmp\_node

\* 連結 cur\_node 與 tmp\_node->next

\* 刪除 tmp\_node

\*

\* EXCAPTION

\* 1.node\_num <= 0 處理：跳出函式

\* 2.node\_num > list長度L

\* 3.node\_num = 1 處理：tmp = head, head = head->next

\* 4.node\_num = 1 且 節點數只有1 處理：同上，head = head->next (= NULL)

\*/

if (node\_num <=0 ) {

cout << "m <=0" << endl;

return;

}

if (node\_num ==1 ) {

node\* tmp\_node = head\_node;

head\_node = head\_node->next();

delete tmp\_node;

return;

}

node\* cur\_node = head\_node;

int i = 1;

while(cur\_node->next()!=NULL && i !=node\_num-1 ){

cur\_node = cur\_node->next();

i++;

}

if (cur\_node->next()==NULL && i != node\_num-1 ) {

cout << "Wrong input, m > list long L" << endl;

return;

}

node\* tmp\_node = cur\_node->next();

cur\_node->next(cur\_node->next()->next());

delete tmp\_node;

}

## HW4\_Q6

##### The Question

Given an implementation of a singly linked list support prepend, append, insert, and delete operations in C++ (This is handwriting homework. All of them are short enough)  
Note : prepend : Insert a node before the first node  
Append : Insert a node after the last node  
Insert : Insert a node before m-th node where m > 1

##### My Answer

// HW4\_Q6

#include <iostream>

using namespace std;

class node {

private:

node \*\_next;

int \_data;

public:

node(){

\_next = NULL;

\_data = 0;

};

node \*next(){return \_next;};

int data(){return \_data;}

void next(node\* n){\_next = n;};

void data(int d) {\_data = d;};

};

class singly\_linked\_list {

private:

node\* head\_node;

public:

void prepend(int data) {

node\* new\_node = new node();

new\_node ->data(data);

new\_node ->next(head\_node);

head\_node = new\_node;

};

void append(int data) {

node\* cur\_node = head\_node;

node\* new\_node = new node();

new\_node ->data(data);

while(cur\_node->next()!=NULL){

cur\_node = cur\_node->next();

}

cur\_node->next(new\_node);

};

void insert(int node\_num, int data) { //node\_num = m

if (node\_num <=0 ) {

cout << "m <=0" << endl;

return;

}

node\* cur\_node = head\_node;

node\* new\_node = new node();

new\_node ->data(data);

int i = 0;

while(cur\_node->next()!=NULL || i !=node\_num ){

cur\_node = cur\_node->next();

i++;

}

if (cur\_node->next()==NULL && i !=node\_num ) {

cout << "Wrong input, m > list long L" << endl;

return;

}

new\_node->next(cur\_node->next());

cur\_node->next(new\_node);

};

void delete\_node(int node\_num) { //node\_num = m

if (node\_num <=0 ) {

cout << "m <=0" << endl;

return;

}

if (node\_num ==1 ) {

node\* tmp\_node = head\_node;

head\_node = head\_node->next();

delete tmp\_node;

return;

}

node\* cur\_node = head\_node;

int i = 1;

while(cur\_node->next()!=NULL && i !=node\_num-1 ){

cur\_node = cur\_node->next();

i++;

}

if (cur\_node->next()==NULL && i != node\_num-1 ) {

cout << "Wrong input, m > list long L" << endl;

return;

}

node\* tmp\_node = cur\_node->next();

cur\_node->next(cur\_node->next()->next());

delete tmp\_node;

};

};

int main(int argc, char const \*argv[]) {

return 0;

}

## HW4\_Q7

##### The Question

Reading assignment: Section 4.3  
What is iterator, template in class ? What are advantages using them ? You can use examples to explain

##### My Answer

iterator : 迭代器

在設計程式時，我們經常會需要將一部分的資料依序取出。像Array、Set，這概念入門者可能比較難理解。  
舉個例子，就像FOR迴圈與Array，我們經常會需要去分Array中析每一筆資料，或對其進行操作，此時我們就會用FOR迴圈逐一對其進行操作。  
但，如果遇到了其他資料型態怎麼辦呢，所以我們通常會去設計一個通用的方法(多型)去逐一取得資料。  
在大部分中的物件導向語言中，通常會內建iterator。  
例如：JAVA的Iterator物件中的next()以及hasNext()、Python的iter()

## HW4\_Q8

##### The Question

How to insert a node before the first node of a circular list?

##### My Answer

void append(int data) {

/\*\*

\* TODO

\* @var cur\_node 存放當前節點

\* @var new\_node 愈加入之新節點

\* 令 cur\_node = head\_node

\* cur\_node 不斷往下一個節點走(while)

\* new\_node 與 cur\_node->next 連接

\* head\_node與 new\_node->next 連接

\*/

}

## HW4\_Q9

##### The Question

Implement Q8 in C++ (This is handwriting homework.)

##### My Answer

// HW4\_Q9

#include <iostream>

using namespace std;

class node {

private:

node \*\_next;

int \_data;

public:

node(){

\_next = NULL;

\_data = 0;

};

node \*next(){return \_next;};

int data(){return \_data;}

void next(node\* n){\_next = n;};

void data(int d) {\_data = d;};

};

class circular\_singly\_linked\_list { //單向環狀鍵結陣列

private:

node\* head\_node;

public:

void append(int data) {

node\* cur\_node = head\_node;

node\* new\_node = new node();

new\_node ->data(data);

while(cur\_node->next()!=head\_node){

cur\_node = cur\_node->next();

}

cur\_node->next(new\_node);

}

}

## HW4\_Q10

##### The Question

What are differences between arrays and pointers in C ?

##### My Answer

Pointer is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before using it to store any variable address. (取至tutorialspoint)  
簡而言之，pointer種存放記憶體位址的變數。  
而array是一連串的記憶體，常見的讀取方法是 a[i] 可以讀寫矩陣中第i個位子，a則是指向這一連串記憶體第一個位址的pointer，而a[i]其實是對a+i位址的記憶體進行存取。

在剛開始學C時，許多人都會犯的一個嚴重錯誤

char\* str="Hello";

str = "BUG"

這段程式碼看起來正常，我們也對str有讀取的權限，但我們竟不能將"BUG"寫入？其實"Hello"已經在記憶體中宣告為一連串的char常數了，我們只是將其所在位址指向str，透過指標只可以去讀該字串常數的內容，而不能寫入。(C 十戒之四)  
更多有關指標操作的範例錯誤可以參考 C 語言新手十誡  
<https://www.ptt.cc/bbs/SFFamily/M.1134111686.A.5F8.html>