S T L

Doraemon 6/6

STL？能吃嗎：D？

來回想一下list ...

- 加入一個結點 ....
- `Node *new_node = new Node();`
- `new_node->next = head->next;`
- `head->next = new_node;`

想想動態記憶體配置...

- `int *new_array;`
- `new_array = new int[100];`
- `delete [] new_array;`

嘿～排序～

- for(int i = 0; i < n; ++i)
  - for(int j = 1; j < n; ++j)
    - if (s[j] < s[j-1]){
      - tmp = s[i];
      - s[i] = s[j];
      - s[j] = tmp;
    - }

~文字轉轉轉~

- char select[100], word[100];
- bool visit[100];
- void rotate(int step){
  - if(step == length) { … }
  - else { .. code 好長省略啦～ .. }
- }

難道不能偷懶嗎orz...

偷懶技術概論

- Standard Template Library
- Container
    - Vector, list, string
- Iterator
- Algorithm
    - sort, next_permutation, lower_bound

警告

- 使用此技巧容易使人變得懶散
- 勤勞者請勿過度使用
- 懶散者使用後會變成~~懶惰之神~~
- 使用前請三思

使用前須知

- 要記得 #include 一些東西噢～～～
- STL 屬於 namespace std;
    - 跟cin, cout 一樣，在前面加上std ::
    - 或者加上using namespace std;

C++ string

#include <string>

string

- 宣告

```
#include <string>
using namespace std;
string str1;      //空字串
string str2("bird");
//string str2 = "bird";
string str3(str2);
```

string

- 指定: =

```
str1 = str2;
```

- 比較: ==, !=, <, <=, >, >=

```
if (str1 == str2)
    cout << “str1 == str2";

if (str2 > str3)
    cout << “str2 > str3” ;
```

string

- 串接: +
- str1 = str1 + str2;
- str1 = str1 + " yo”;
- string str4 = "I'm" + " happy.”
- str4 = str4 + " " + "Yo.";

string

- 取值

```
string str1("tree");
cout << str1[0] << " ";
cout << str1[1] << " ";
cout << str1[2] << " ";
cout << str1[3] << " ";
```

string的各種操作

- size(), length()
- string str1(“tree");
- cout << str1.size() << endl;
- cout << str1.length() << endl;

string的各種操作

- 取得子字串

```
substr(pos, n)

substr(pos)

substr()
```

string的各種操作

EX：

```
string str("tree");

cout << str.substr(0, 2) << endl;

cout << str.substr(1) << endl;

cout << str.substr() << endl;
```

string的各種操作

- 尋找

  find(c)

  find(str)

  find(c, pos)

  find(str, pos)

  會回傳找到的第一個位置,

  若找不到會回傳string::npos,

  npos為"絕對比任何有效索引還大"的值

string的各種操作

Ex:

```
string str("I love the tree.");

cout << str.find(“ee") < <endl;

cout << str.find('t', 8) << endl;

cout << str.find("love", 6) << endl;
```

~~伸縮自如~~的陣列

Vector

#include <vector>

Vector

- 動態陣列
- 不用自己手動 `new / delete`
- 支援隨機插入，但是很慢，後面會講到`list`
- 支援隨機存取，可以當陣列使用：D

Vector 宣告

- std::vector <資料形態> 變數名稱;
- 資料形態：int, double, string, 自訂形態..
- EX.
  - std::vector < int > V;
  - std::vector < string > V;
  - std::vector < Node > V;

Vector 使用方法

- size() : 回傳 vector 的大小
- clear() : 把陣列裡面的東西清光
- [] -> 中括號 : 跟陣列一樣
  - Ex. V[0], V[1], V[2] . . . .
- push_back(東西) : 在尾巴加入一個東西
- pop_back() : 從尾巴拔掉一個元素

展示 Vector

鏈表

list

#include <list>

list

- 就是你知道的那個寫起來很痛苦的`linked-list`
- 可以隨機插入：D
- 不能隨機存取，不能當陣列用：（

list 宣告

- std::list <資料形態> 變數名稱;
- 資料形態：int, double, string, 自訂形態..
- EX.
  - std::list < int > L;
  - std::list < string > L;
  - std::list < Node > L;

list 用法

- `size()`：回傳裡面有幾個東西（回傳大小）
- `clear()`：清空
- `push_front(東西)`：把東西塞在前面
- `push_back(東西)`：把東西塞在後面
- `pop_front()`：從前面拔掉一個東西
- `pop_back()`：從後面拔掉一個東西

展示 list

迭代器 Iterator

Iterator

- 他看起來就是很像指標
- 可以指到容器(container)的某個元素
- 不用管container到底是怎麼做的

Iterator 宣告

- container <type>::iterator 變數;
- container : vector, list, string ……
- type : int, char, double, 自訂形態...
  - 要和被指到的container的type一樣

Iterator 用法(1)

- EX. std::string str;
  - std::string::iterator iter = str.begin();
  - *iter == str[0];
  - iter++;
  - *iter == str[1];

Iterator 用法(2)

- `std::vector <int> V;`
- `std::vector <int>::iterator iter;`
- `for(iter = V.begin(); iter != V.end(); ++iter)`
  - `std::cout << *iter << std::endl;`

Iterator 用法注意

- 如果你指到的那個位置被刪掉（pop_back(), 之類的
- Iterator 不保證他指到的地方是對的喔
- 參考資料在這裡：
    - http://stackoverflow.com/questions/6438086/iterator-invalidation-rules

Iterator Demo

課堂練習 190

Algorithm

#include <algorithm>

Algorithm

- 裡面有超多好用的工具
- 你只要用過就會上癮（？）

sort

- 對 `int arr[100];`
- `sort(arr, arr + 100);`
- 對 `vector<int> V;`
- `sort(V.begin(), V.end());`

lower_bound

- 找到某個數字的下界（（ 還記得二分搜嗎ＸＤ“～
- 他回傳的是container某個元素的**位置**！！！
- 對 `int arr[100], x = 100;`
- `int *ptr = lower_bound(arr, arr+100, value);`
- 對 `vector<int> V;`
- `vector<int>::iterator lb;`
- `lb = lower_bound(V.begin(), V.end(), x);`
- `int distance = lb - V.begin(); //距離頭多遠`

next_permutation

- 下一個排列
- 一開始請記得把它排好，否則 … 嘿嘿
- next_permutation(begin, end);
- 要用 do {

```
    . . .
    . . .
    . . .
} while(next_permutation(…));
```

Algorithm Demo

好用的工具很多：D

想要知道更多...

請自己Google (X)

課堂練習 261