Introduction & Lab 1

https://reurl.cc/8vxa7b

Outline

- Reminders & Rules
- C++ and STL
  - Brief intro to C++
  - STL (read reference, iterator, vector, string)
- Binary Search

Reminders

- We assume everyone taking this course have learned C++
- Codes in this course uses C++17 standards
  - Use `-std=c++17` flag when compiling
  - We only reply problems with C++
- Online judge accepts C++ / C / python3
  - We only guarantee problems can be solved with C++
- Standard Template Libraries(STL) are allowed in this course
  - <iostream>
  - <algorithm>
  - <vector>
  - …
- You are free to use any C++ compilers, IDEs, editors
  - EECS 328 is installed with codeblocks, vscode, dev C++(?)

Rules -- Public Discussion

- Lectures, labs, coding assignments problems
  - Post them publicly on eeclass discussion board
  - DO NOT send private emails to TAs to ask about lectures, labs, coding, we ignores them
  - TA may refuse to help if your post is harmful or hostile to us
- Private issues (e.g. plagiarism, grades)
  - Send and cc (副本) to all of us to avoid email miss
  - Prof. Hon: wkhon@cs.nthu.edu.tw
  - Prof. Shen: chihya@cs.nthu.edu.tw
  - TA Xavier: xavier0505@gapp.nthu.edu.tw
- If you are competent to answer classmate’s question, Don’t be shy!

Rules -- Coding Problems

- If you need direct assistance, please come to lab
- Please use natural human language to explain your thoughts
- Please add comments in your codes
- Please name the variables properly
- Please paste codes through sites that supports indent and highlights
  - codepad.org
  - gist.github.com
  - Ideone
- DO NOT printscreen, take pictures of your screen, paste raw codes on eeclass
- Plagiarism is definitely not tolerated

Rules -- HW & Exam

- Labs
  - Duration: Until Exams
  - Provide AC code
- Coding Practice
  - Duration: 2 weeks
  - 4 Questions per Assignment
  - You have to finish it by yourself
- Exam
  - Duration: 3 hour
  - 4 Questions
  - Conducted face-to-face at EECS 328(and 326)

Online Judge - HW

https://acm.cs.nthu.edu.tw

try to log in the account

- username: DSH+student id, password: student id
- example: DSH111000104 / 111000104

If you forgot your password or username, please email TAs directly with proper proof of who you are.

Online Judge - Exams

- site and account will be given right before exams.
- You will NOT be able to connect to other sites.
- We will have an announcement before each exams, and the seating arrangements will be attached.

- Any form of Plagiarism is definitely not tolerated(USB, cheat sheet, cell phone, discussion…)
- If you have violated the exam rules, we will flunk you directly.

C++ and STL Intro

Brief Review on C++

- Header file
- using namespace std
- main function
- cin, cout

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

int main() {
  int a, b;
  cin >> a >> b;
  cout << a + b << "\n";
}
```

Typically,

- `int` can represent numbers from $-2^{31} \sim 2^{31}-1$
- `long long int` represent numbers from $-2^{63} \sim 2^{63}-1$
- use `double` instead of `float`
- Be careful of overflow

Brief Review on C++: reference (參照)

```
void swap(int &a, int &b) {
    int tmp = a;
    a = b, b = tmp;
}

int main() {
    int a = 487, b = 63;
    swap(a, b);  // a == 63, b == 487
}
```

Brief Review on C++: reference (參照)

```
int upper_bound(vector<int> &arr, int n, int val) {
    int l = 0, r = n-1, mid, ret = -1
    while (l <= r) {
        mid = (l+r)/2;
        if (arr[mid] > val) ret = mid, r = mid-1;
        else l = mid+1;
    }
    return ret;
}
```

Brief Review on C++: STL

C++ STL (standard library) provides a wide range of containers and algorithms

Reference: https://en.cppreference.com/w/

Containers:

- std::vector
- std::string
- std::set
- std::queue
- …

Algorithms:

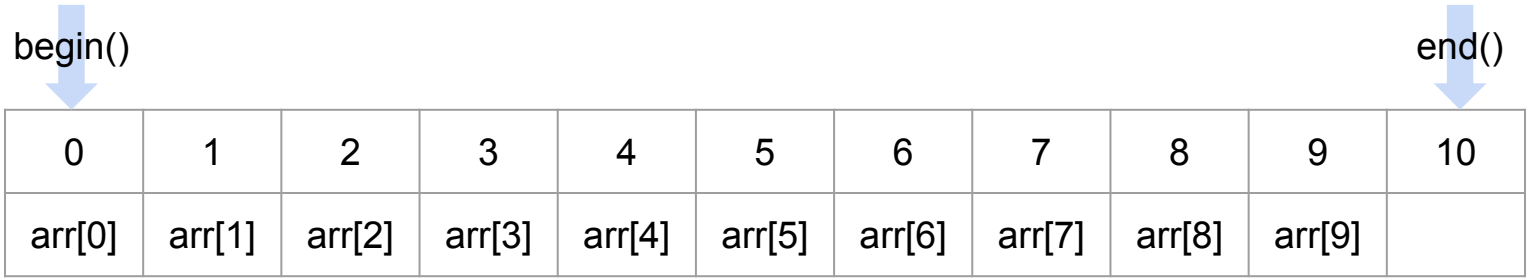
- std::sort
- std::binary_search
- std::lower_bound
- std::upper_bound
- …

Brief Review on C++: STL

When using STL, please notice:

- Template parameters
- Constructors (STL container only)
- Function parameters
- Return type
- Time complexity

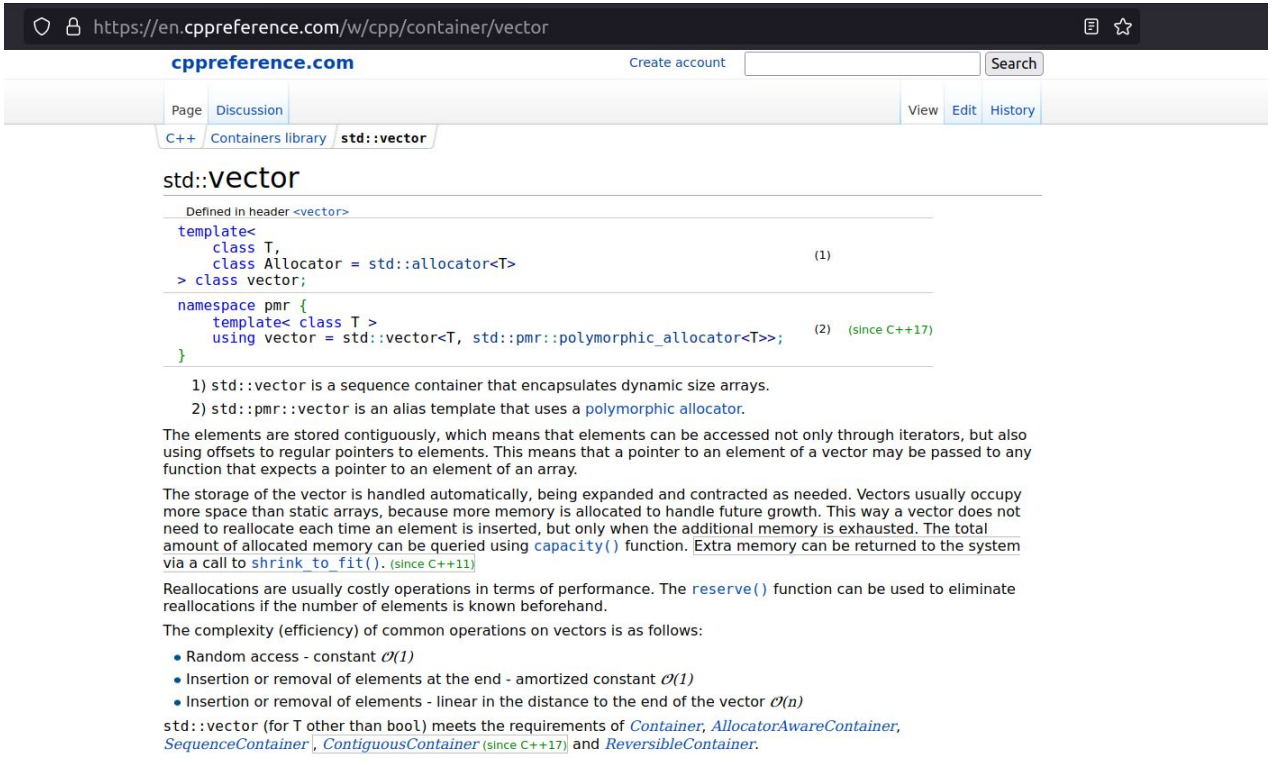
Iterator




```
int main() {
  vector<int> arr(10, 0);
  vector<int>::iterator L = arr.begin();
  vector<int>::iterator R = arr.end();
}
```

std::vector @

A dynamic array

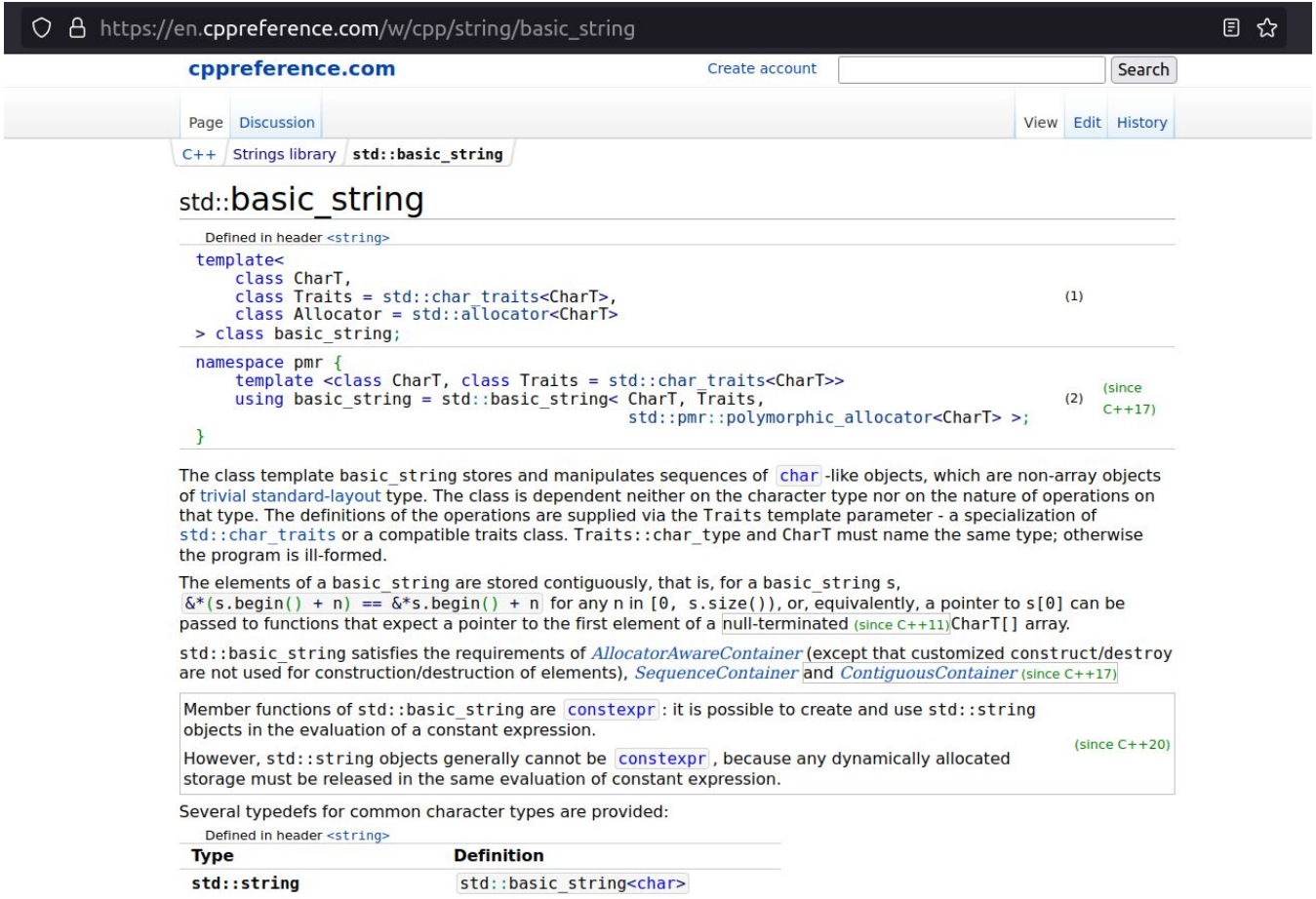


std::vector

- Variable size array
  - O(1) insert/delete element in the back
  - O(n) insert/delete arbitrary element
- O(1) random access


```
vector<int> arr(100, 0);
arr.emplace_back(87);
arr.pop_back();
arr[12] = 34;
sort(arr.begin(), arr.end());
```

std::string @



https://en.cppreference.com/w/cpp/string/basic_string

cppreference.com Create account Search

Page Discussion View Edit History

C++ Strings library std::basic_string

std::basic_string

Defined in header <string>

```
template<
    class CharT,
    class Traits = std::char_traits<CharT>,
    class Allocator = std::allocator<CharT>
> class basic_string;
```
(1)

```
namespace pmr {
    template <class CharT, class Traits = std::char_traits<CharT>>
    using basic_string = std::basic_string< CharT, Traits,
                                            std::pmr::polymorphic_allocator<CharT> >;
}
```
(2) (since C++17)

The class template `basic_string` stores and manipulates sequences of `char`-like objects, which are non-array objects of trivial standard-layout type. The class is dependent neither on the character type nor on the nature of operations on that type. The definitions of the operations are supplied via the `Traits` template parameter - a specialization of `std::char_traits` or a compatible traits class. `Traits::char_type` and `CharT` must name the same type; otherwise the program is ill-formed.

The elements of a `basic_string` are stored contiguously, that is, for a `basic_string s`, `&*(s.begin() + n) == &*s.begin() + n` for any `n` in `[0, s.size())`, or, equivalently, a pointer to `s[0]` can be passed to functions that expect a pointer to the first element of a null-terminated (since C++11) `CharT[]` array.

`std::basic_string` satisfies the requirements of *AllocatorAwareContainer* (except that customized `construct`/`destroy` are not used for construction/destruction of elements), *SequenceContainer* and *ContiguousContainer* (since C++17)

Member functions of `std::basic_string` are `constexpr`: it is possible to create and use `std::string` objects in the evaluation of a constant expression.

However, `std::string` objects generally cannot be `constexpr`, because any dynamically allocated storage must be released in the same evaluation of constant expression.

(since C++20)

Several typedefs for common character types are provided:

Defined in header <string>

Type	Definition
std::string	`std::basic_string<char>`

std::string

C++ STL 提供的字串類別比傳統的 char[] 方便許多

```
string str = "I_am";
str += "_string";
str[1] = str[4] = ' ';
cout << str; // "I am string"
```

*不是用 '\0' 當結尾

struct / std::pair / std::tuple

e.g. Representing a 2D-coordinate?

```
// arrays
int x[10], y[10];
x[0] = 1;
y[0] = 1;
```

```
// std::pair
pair<int,int> arr[10];
arr[0].first = 1;
arr[0].second = 2;
```

```
// struct or class
struct Point { int x, y; };
Point arr[10];
arr[0].x = 1;
arr[0].y = 2;
```

```
// std::tuple
tuple<int,int> arr[10];
get<0>(arr[0]) = 1;
get<1>(arr[0]) = 2;
```

Typename Alias、auto

- Replace lengthy type name with using-declaration
- Increase readability with “auto”

```
using Iter = vector<int>::iterator;
vector<int> arr(487, 63);
vector<int>::iterator it1 = arr.begin();
Iter it2 = arr.begin();
auto it3 = arr.begin();          // c++11

tuple<int,int,int> point3D;
auto &[x, y, z] = point3D;       // c++17
```

Range Based For (C++11, C++17 or later)

```
vector<int> arr(487, 63);
for (auto &x : arr) {
  cout << x << "\n";
}

map<int,int> mp;
for (auto &[key, val] : mp) {
  cout << key << "," << val << "\n";
}
```

Learning Resources

1. C++ Language
   a. https://en.cppreference.com/w/
   b. https://cplusplus.com/reference/
   c. <<A tour of C++>> (@)
2. Algorithm & Data Structures
   a. <<Introduction to Algorithms>>
   b. <<Fundamentals of Data Structures in C++>>

Most of them are beyond the scope of this course :)

Binary Search

What is Binary Search?

- Let’s play a game first
  - Pick a number in range [1, 100]
  - Can you guess what’s the number I picked?

- How to play it optimally?
  - How many guesses will it take to get the correct number?
  - For worst case, how many guesses?

This is Binary Search!!!

Binary Search on Arrays

- Given an Array of Integers

index	0	1	2	3	4	5	6	7
value	33	16	2	10	8	57	26	13

- Determine whether certain value is in array

Binary Search on Arrays

- Given an Array of Sorted Integers

index	0	1	2	3	4	5	6	7
value	2	8	10	13	16	26	33	57

- Determine whether certain value is in array

What is Binary Search?

- Let's say the range is [1, N](or [0, N - 1])
  - How many guesses it takes?
  - What's the upper bound for worst case?

- It takes O(logN) to find the answer!!!

- Which means in no more than ($\log_2 N$) times, you must get the answer

When to use Binary Search?

- The problem must satisfy some Monotonicity
  - Increasing, Decreasing, Non-increasing, Non-decreasing

- No only for determining whether certain value exists, you can also…
  - lower_bound()
  - upper_bound()
  - …

- Also, there are a lot more problems that the concept of Binary Search will come in handy.

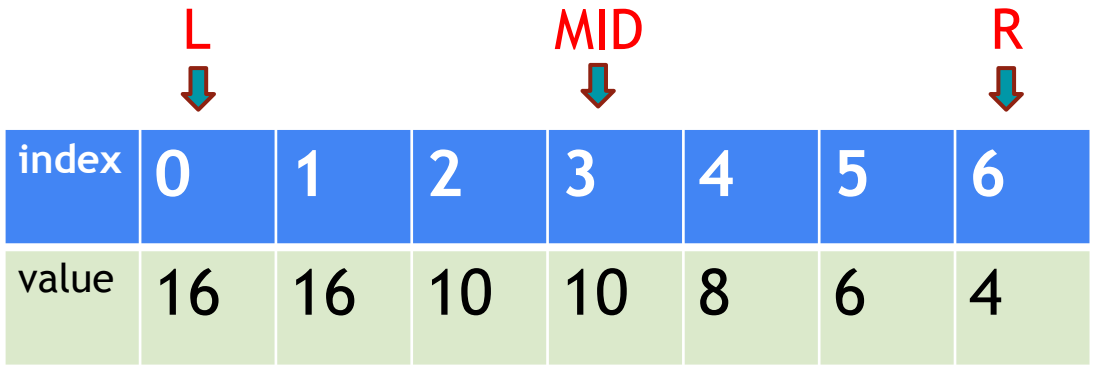
Binary Search Problem

- Given a **non-increasing** integer array A with ***n*** integers.
- Answer ***m*** queries.

Restrictions

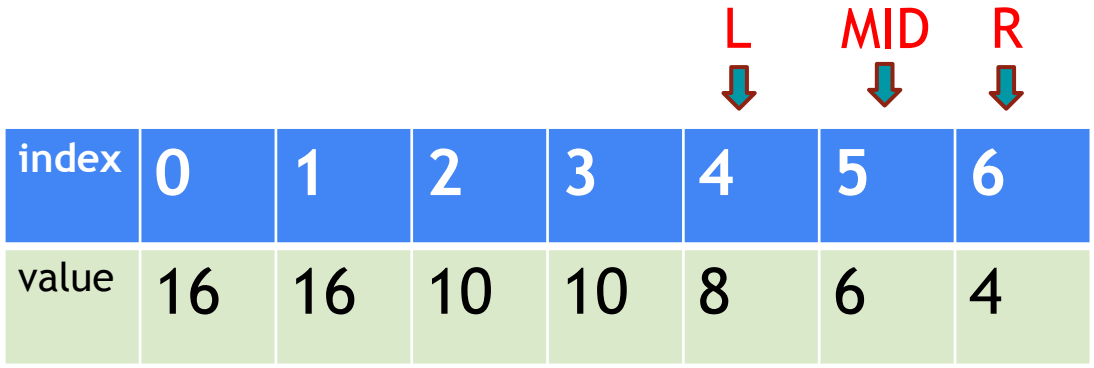
- $1 \leq n \leq 10^5, 1 \leq m \leq 10^5$
- $-10^{16} \leq a_i \leq 10^{16}$ for $i = 1, 2, \cdots, n$
- $-10^{16} \leq q_i \leq 10^{16}$ for $i = 1, 2, \cdots, m$

Binary Search (Implement)




```cpp
bool my_binary_search(vector<LL> &arr, LL query)
{
    int l = 0, r = arr.size() - 1, mid;
    while (l <= r)
    {
        mid = (l + r) / 2;
        if (arr[mid] == query) return true;
        else if (arr[mid] > query) l = mid + 1;
        else r = mid - 1;
    }
    return false;
}
```

Binary Search (Implement)




```cpp
bool my_binary_search(vector<LL> &arr, LL query)
{
    int l = 0, r = arr.size() - 1, mid;
    while (l <= r)
    {
        mid = (l + r) / 2;
        if (arr[mid] == query) return true;
        else if (arr[mid] > query) l = mid + 1;
        else r = mid - 1;
    }
    return false;
}
```

Class Implementation

Binary Search:

https://ideone.com/dRmQJX

Time Complexity

- OJ上一秒大約可以跑10^9個指令
- 將測資大小帶入複雜度中即可估算程式會不會TLE
- 練習？

常見時間複雜度與適用題目大小

複雜度	題目大小
$O(n!)$	$n \leq 10$
$O(2^n)$	$n \leq 20$
$O(n^4)$	$n \leq 50$
$O(n^3)$	$n \leq 200$
$O(n^2)$	$n \leq 3000$
$O(n \lg n)$	$n \leq 10^6$