Bit Set:

1. Cách Import?

#include <bitset>

1. Đối Tượng Bit Set?

* Tạo 1 Bit Set

std::bitset<<Số Bit>> <Bit Set>;

* Kích thước bộ nhớ cấp phát cho <Bit Set> là 4, 8, 16, … Byte tùy thuộc vào

<Số Bit>, nó sẽ chọn kích thước nhỏ nhất sao cho không nhỏ hơn <Số Bit>

* Ví dụ

std::bitset<10> foo;

* Giá trị mặc định của <Bit Set> là tất cả các Bit = 0
* Để tạo giá trị ban đầu

std::bitset<<Số Bit>> <Bit Set>(<String Nhị Phân>);

* <String Nhị Phân> sẽ chỉ lấy <Số Bit> kí tự đầu tiên, mấy kí tự đằng sau không quan tâm, <Số Bit> kí tự đầu tiên phải là 0 hoặc 1
* Nếu <String Nhị Phân> có số kí tự nhỏ hơn <Số Bit>, thì nó sẽ bù vài số 0 vào bên trái
* <String Nhị Phân> cũng có thể là số nguyên hoặc Floating Point, nếu Floating Point sẽ được chuyển về số nguyên, sau đó sẽ chuyển sang dạng nhị phân của chúng
* Ví dụ

std::bitset<8> foo("10111010");

std::bitset<8> bar('g');

* Thay đổi 1 Bit

<Bit Set>[<Index>] = <0 Hoặc 1>;

* <Index> được đánh từ phải qua trái, bắt đầu từ 0
* Ví dụ

foo[1] = 0;

* Ta có

|  |  |
| --- | --- |
| foo trước | 11111010 |
| foo sau | 11111000 |

* Copy giá trị của 1 Bit Set cho 1 Bit Set

<Bit Set Đích> = <Bit Set Mà Giá Trị Của Nó Bị Copy>;

C Math:

1. Cách Import?

#include <cmath>

1. Tính Lũy thừa?

pow(<Cơ Số>, <Số Mũ>);

* Ví dụ

foo = pow(2.5, 3.5);

* Ta có

|  |  |
| --- | --- |
| foo | 24.7053 |

Input Output Stream:

1. Cách Import?

#include <iostream>

#include <iomanip>

1. Chèn Các Kí Tự Vào Buffer?

std::cout << <Dòng Chữ>;

* <Dòng Chữ> sẽ được đẩy vào Buffer được cout chõ đến
* Lưu ý “<<” là toán tử dịch Bit sang trái nhưng đã bị viết đè lên, độ ưu tiên = độ ưu tiên của toán tử dịch Bit, giá trị trả về cũng chính là đối tượng std::cout luôn
* Ví dụ

std::cout << "Hello";

* Chèn kí tự xuống dòng

std::cout << std::endl;

1. Cách Tạo User Input?

std::cin >> <Biến 1> >> <Biến 2> >> …;

* Lệnh trên tương đương

scanf("%<Kiểu Dữ Liệu 1>%<Kiểu Dữ Liệu 2>…", &<Biến 1>, &<Biến 2>, …);

* Ví dụ

int foo;

float bar;

char alice[30];

std::cin >> foo >> bar >> alice;

* Tương đương

scanf("%d%f%s", &foo, &bar, alice);

* Lưu ý “>>” là toán tử dịch Bit sang phải nhưng đã bị viết đè lên
* Tuy nhiên, lệnh trên sử dụng Buffer của cin, không phải của scanf, trừ khi chúng dùng chung như khi ở chế độ mặc định
* Đồng thời, cin sẽ có các cờ lỗi cho riêng nó, gồm good, eof, fail và bad
* Trả về giá trị cờ eof

std::cin.eof()

* Cờ này sẽ được kích hoạt khi bạn cố tình chỉ gán chuỗi Bit EOF cho 1 biến, bằng cách nhấn “Ctrl” + “Z” sau đó nhập chuỗi bất kì đằng sau, rồi nhấn “Enter”, tổng cộng chuỗi vừa nhập không được thêm vào Buffer của cin
* Trả về giá trị cờ fail

std::cin.fail()

* Cờ này sẽ được kích hoạt khi không thể trích xuất giá trị trong Buffer của cin để gán cho biến, ví dụ trong Buffer chứa abc, mà biến foo của bạn kiểu int, thì khi này, không có Bit nào trong Buffer được trích xuất cho foo, ví dụ nếu là 123abc thì còn trích xuất được 123, đằng này không trích xuất được gì hết, và khi không trích xuất được gì hết, cờ fail sẽ kích hoạt
* Trả về giá trị cờ bad

std::cin.bad()

* Cờ này chỉ kích hoạt khi có lỗi hệ thống trong quá trình trích xuất dữ liệu từ cin
* Trả về giá trị cờ good

std::cin.good()

* Cờ này chỉ kích hoạt khi tất cả cờ còn lại không kích hoạt, và không kích hoạt khi chỉ cần 1 cờ trong các cờ còn lại kích hoạt
* Khi không thể trích xuất được gì để gán cho biến, tất cả Bit của biến đều thành 0
* Khi cờ good không kích hoạt, lệnh std::cin >> <Biến> sẽ bị bỏ qua
* Khi bị lỗi, thì nội dung nhập còn lại vẫn được lưu trong Buffer của cin
* Để kích hoạt lại cờ good và bất hoạt các cờ còn lại, Buffer vẫn giữ nguyên

std::cin.clear();

* Để trích xuất từ vị trí hiện tại trong Buffer đến vị trí của kí tự <Kí Tự> đầu tiên

std::cin.getline(<Biến>, <Kí Tự>);

* Tương đương lệnh sau, chỉ khác là dùng Buffer của cin

scanf("%[^<Kí Tự>]", &<Biến>);

* Để xóa kí tự đầu tiên ra khỏi Buffer của cin

std::cin.ignore();

* Tương đương
* Để xóa N kí tự kể từ kí tự đầu tiên ra khỏi Buffer của cin

std::cin.ignore(<N>);

* Để xóa các kí tự kể từ kí tự đầu tiên, cho tới khi xóa được N kí tự hoặc gặp

<Kí Tự> đầu tiên, ra khỏi Buffer của cin, <Kí Tự> cũng sẽ bị xóa

std::cin.ignore(<N>, <Kí Tự>);

1. Tạm Dừng Chương Trình Cho Đến Khi Ấn 1 Phím Bất Kì?

system("pause");

* Màn hình Console sẽ chèn thêm dòng chữ sau

Press any key to continue . . .

1. Tạo Không Gian Cho Chữ In Ra?

std::cout << std::setw(<Độ Rộng>);

* Lệnh trên sẽ kích hoạt cờ setw cho đối tượng cout, cờ này làm cho thứ tiếp theo được đẩy vào Buffer sẽ sở hữu phần không gian = <Độ Rộng>, là 1 số nguyên chỉ định số kí tự có thể chứa trong không gian này, đồng thời sẽ bất hoạt cờ này ngay sau đó
* Ví dụ

std::cout << std::setw(12) << “Dick” << “Head”;

* Thứ được thêm vào Buffer

DickHead

* <Độ Rộng> mà < số kí tự tiếp theo được thêm vào Buffer thì sẽ tăng = số kí tự tiếp theo được thêm vào Buffer
* Mặc định cout sử dụng cờ right có tác dụng căn phải chữ trong vùng không gian nó chiếm, để chuyển sang sử dụng cờ left có tác dụng căn trái

std::cout << std::left;

* Để chuyển sang cờ right có tác dụng căn phải

std::cout << std::right;

1. Chỉ Định Số Chữ Số Thập Phân Được In Ra?

* Đối tượng cout mặc định không dùng cờ fixed, nên khi chèn số Floating Point vào Buffer nó chĩa vào, thì số này sẽ tự động làm tròn 6 chữ số tính từ số đầu tiên bên trái cùng ở phần nguyên, nếu phần nguyên vượt quá 6 số thì hiện = kí hiệu khoa học với Mantissa chỉ gồm 6 số
* Để cout sử dụng cờ fixed, khi đó thay vì tính từ số đầu tiên phần nguyên = tính từ số đầu tiên phần thập phân, đồng thời nếu phần thập phân ít số thì bù 0

std::cout << std::fixed;

* Mặc định cout sẽ sử dụng cờ precision với giá trị 6, do đó số chữ số được làm tròn là 6 như đã nói ở trên, để thay đổi cờ precision của cout thành 1 số nào đó

std::cout << std::setprecision(<Số Chữ Số>);

* Hoặc

std::cout.precision(<Số Chữ Số>);

* <Số Chữ Số> là số nguyên không âm
* Để trả về <Số Chữ Số> hiện tại của cout

std::cout.precision()

* Ví dụ

std::cout << std::setprecision(4) << 123.467 << std::endl;

std::cout << std::fixed << std::setprecision(5) << 123.467 << std::endl;

* Màn hình CMD

123.5

123.46700

1. Chuyển Sang Chế Độ In Ra Kí Hiệu Khoa Học?

* Mặc định cout không sử dụng cờ scientific, để kích hoạt cờ này cho cout

std::cout << std::scientific;

1. Chuyển Sang Chế Độ In Luôn Cả Dấu + Đằng Trước Số Dương?

* Mặc định cout sử dụng cờ noshowpos không thêm dấu + đằng trước số dương, để chuyển sang sử dụng cờ showpos, có tác dụng thêm 1 dấu + đằng trước số dương, gồm cả nguyên và Floating Point

std::cout << std::showpos;

* Để chuyển lại sử dụng noshowpos

std::cout << std::noshowpos;

* Ví dụ

std::cout << std::showpos << 123.45 << std::endl << -5;

* Thứ được thêm vào Buffer

+123.45

-5

1. Chuyển Sang Chế Độ In Ra Dưới Dạng Bát Phân, Thập Lục Phân?

* Bát phân, chuyển sang sử dụng cờ oct cho cout

std::cout << std::oct;

* Thập lục phân, chuyển sang sử dụng cờ hex cho cout

std::cout << std::hex;

* Trở về thập phân, chuyển về mặc định, sử dụng cờ dec

std::cout << std::dec;

* Khi chèn vào Buffer sẽ không có Prefix 0, 0x, …

1. Kích Hoạt Nhiều Cờ 1 Lúc Cho Cout?

std::cout << std::setiosflags(std::ios::<Tên Cờ 1> | std::ios::<Tên Cờ 2> | …);

* Ví dụ

std::cout << std::setiosflags(std::ios::fixed | std::ios::showpos | std::ios::left);

std::cout << std::setw(15) << 123.45 << "foo";

* Thứ được thêm vào Buffer

+123.450000 foo

1. Lưu Tất Cả Chế Độ Và Reset Chế Độ?

* Để lưu lại trạng thái các cờ của đối tượng cout, có thể sử dụng với đối tượng khác

std::ios::fmtflags <Bản Lưu Chế Độ> = std::cout.flags();

* Các chế độ được lưu bao gồm fixed, left, right, hex, dec, oct, …
* Sử dụng trong trường hợp ví dụ ta muốn bây giờ fixed, nhưng sau đó thì không fixed, …
* Để khiến cout sử dụng lại các cờ cũ đã lưu

std::cout.flags(<Bản Lưu Chế Độ>);

1. Đối Tượng String?

std::string <String> = <Chuỗi>;

* <Chuỗi> phải được đặt trong dấu nháy kép
* Ví dụ

std::string foo = "DickHead";

* <String> là 1 đối tượng của Class std::string, do đó nó sẽ sử dụng được các phương thức trong Class này được lưu ở phần nhớ Text, đồng thời chứa Pointer đến mảng Char tương ứng trong Heap
* Các kí tự trong <Chuỗi> chiếm các Byte liên tiếp nhau theo đúng thứ tự chiều tăng dần địa chỉ
* Để tạo 1 String khác giống như String ban đầu nhưng không dùng chung mảng Char trong Heap

<String Copy> = <String Gốc>;

* Cũng như trên, nhưng chỉ Copy 1 phần

<String Con> = <String Gốc>.substr(<Index Bắt Đầu>, <Số Kí Tự>);

* Hoặc

std::string <String Con>(<String Gốc>, <Index Bắt Đầu>, <Số Kí Tự>);

* Nếu không chỉ định <Số Kí Tự> thì lấy hết
* Ví dụ

std::string bar = foo.substr(2, 4);

* Ta có

|  |  |
| --- | --- |
| bar | "ckHe" |

* Để trả về số kí tự của 1 String

<Số Kí Tự> = <String>.size();

* Để trả về mảng Char tương ứng trong Heap

<String>.c\_str()

* Trả về biểu diễn String của 1 số

std::to\_string(<Số>)

* Để trả về kiểu số nguyên của 1 String

std::stoi(<String>)

1. Chuyển Sang Chế Độ Cout Và Cin Dùng Buffer Riêng So Với Printf Và Scanf?

std::ios\_base::sync\_with\_stdio(false);

* Một khi đã chuyển sang chế độ này thì không thể trở về chế độ mặc định

1. Gỡ Ràng Buộc Cin Với Cout?

std::cin.tie(NULL);

* Để ràng buộc lại

std::cin.tie(&std::cout);

1. Trả Về Pointer Vào Phần Tử Đầu Tiên Hoặc Kế Ngay Sau Cuối Cùng Của 1 Mảng?

<Pointer Đầu> = std::begin(<Mảng>);

<Pointer Kế Sau Cuối> = std::end(<Mảng>);

* Ví dụ mảng int [1, 2, 3], thì giả sử địa chỉ của phần tử đầu là 100 thì địa chỉ kế sau cuối sẽ là 112, theo hệ thập phân

1. Tạo 1 Đối Tượng Mới Giống Cout?

std::ostream <Đối Tượng Stream Mới>(std::cout.rdbuf());

* Ở đây lệnh rdbuf sẽ trả về con trỏ tới Buffer của đối tượng gọi nó, ở đây là cout, kiểu dữ liệu là std::streambuf\*
* cout bản chất là 1 đối tượng thuộc Class ostream tức là Output Stream
* <Đối Tượng Stream Mới> dùng Buffer mà cout tại thời điểm gọi lệnh trên chĩa vào, nó có các cờ riêng biệt so với cout gốc, các cờ này ban đầu ở trạng thái giống cout mặc định
* Ví dụ

std::cout << std::fixed << std::setprecision(5) << 12.345 << std::endl;

std::ostream foo(std::cout.rdbuf());

foo << 12.345;

* Thứ được thêm vào Buffer mà cout và foo chĩa vào

12.34500

12.345

1. Thay Đổi Buffer Mà Cout Chĩa Vào?

std::cout.rdbuf(<Đối Tượng Stream>.rdbuf());

* <Đối Tượng Stream> có thể là 1 File Object, khi đó, những bạn có thể dùng cout để ghi vào Buffer của File tương tự như khi dùng File Object đó

1. Trả Về 1 Số Ngẫu Nhiên?

rand()

* Lệnh trên sẽ trả về 1 số nguyên ngẫu nhiên trong đoạn từ 0 đến 32767, tức là từ 0 đến số nhị phân 15 Bit Full 1
* Các số ngẫu nhiên sẽ được tạo ra dựa vào Seed, nếu Seed không đổi thì mỗi lần chạy lại chương trình lệnh trên sẽ trả về chuỗi số giống nhau
* Mặc định Seed = 1
* Để chỉ định Seed

srand(<Seed>);

* <Seed> phải là số nguyên không dấu

1. Một Số Giá Trị Đặc Biệt?

NULL = 0

* Đây là địa chỉ, chỉ được phép gán cho Pointer

EOF = -1

1. Một Số Kiểu Dữ Liệu Rút Gọn?

time\_t = ptrdiff\_t = long long

C String:

1. Cách Import?

#include <cstring>

1. Copy 1 Chuỗi Các Ô Byte Ở Vị Trí Này Vào Vị Trí Khác?

strcpy(<Mảng Char/Char Pointer Đích>, <Mảng Char/Char Pointer Nguồn>);

* Copy từng Byte hay kí tự một, cho đến khi gặp kí tự kết thúc là ‘\0’ thì dừng lại, ‘\0’ cũng được Copy sang
* Ví dụ

strcpy(foo, bar);

* Để Copy chỉ K ô, thì dùng lệnh

strncpy(

<Mảng Char/Char Pointer Đích>,

<Mảng Char/Char Pointer Nguồn>,

<K>

);

* Nếu Copy các kí tự sau kí tự ‘\0’ của nguồn, thì các kí tự này cũng = ‘\0’

1. Trả Về Chiều Dài Chuỗi Các Ô Byte Cho Tới Khi Gặp Ô Byte = 0?

strlen(<Char Pointer>)

* Hàm trên sẽ đếm số ô Byte kể từ ô được <Char Pointer> chĩa vào, cho đến khi gặp ô ‘\0’
* Ví dụ
* Cho các ô Byte lần lượt là

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 100 | 101 | 102 | 103 | 104 |
| a | b | c | d | \0 |

* foo là Char Pointer chĩa vào 100, thì khi này strlen(foo) sẽ trả về 4

1. Trả Về Index Của 1 Chuỗi Ô Byte Con Trong 1 Chuỗi Ô Byte Cha?

strstr(<Char Pointer Cha>, <Char Pointer Con>)

* Lệnh trên sẽ trả về Char Pointer tới ô Byte đầu tiên
* Ví dụ

char \* bob = strstr(foo, bar);

* Ta có
* foo

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 |
| a | b | c | d | e | c | d | e | /0 |

* bar

|  |  |  |  |
| --- | --- | --- | --- |
| 102 | 103 | 104 | 105 |
| c | d | e | /0 |

* Thì bob sẽ chĩa tới ô 102
* Nếu không tồn tại chuỗi con trong chuỗi cha, thì trả về Null Pointer

1. Trả Về Index Của 1 Ô Byte?

strchr(<Char Pointer Cha>, <Kí Tự>)

* Lệnh trên sẽ trả về Char Pointer tới ô Byte đầu tiên chứa <Kí Tự>
* Ví dụ

char \* bob = strstr(foo, 'd');

* Ta có
* foo

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 |
| a | b | c | d | e | c | d | e | /0 |

* Thì bob sẽ chĩa tới ô 103

1. So Sánh 2 Chuỗi?

<Lớn Bé Bằng> = strcmp(<Char Pointer 1>, <Char Pointer 2>);

* <Lớn Bé Bằng> = 1 khi chuỗi ứng với <Char Pointer 1> lớn hơn chuỗi ứng với <Char Pointer 2>, = –1 khi bé hơn, = 0 khi bằng nhau

Type Information – Thông Tin Kiểu:

1. Cách Import?

#include <typeinfo>

#include <iostream>

1. Xem Kiểu Dữ Liệu Của 1 Biến?

const std::type\_info &<Thông Tin Kiểu> = typeid(

<Biến Hoặc Giá Trị Hoặc Kiểu Dữ Liệu>

);

* Để xem tên kiểu dữ liệu

const char \*<Kiểu Dữ Liệu> = <Thông Tin Kiểu>.name();

* Các giá trị tương ứng

|  |  |
| --- | --- |
| <Kiểu Dữ Liệu> | Kiểu tương ứng |
| "i" | int |
| "j" | unsigned |
| "l" | long |
| "s" | short |
| "f" | float |
| "d" | double |
| "c" | char |
| "b" | bool |
| "Ss" | std::string |
| "y" | unsigned long long |
| "x" | long long |
| "m" | unsigned long |
| "t" | unsigned short |
| "h" | unsigned char |
| "a" | signed char |
| "e" | long double |
| "Pi" | int\* |

* Chương trình nhanh

#include <typeinfo>

#include <iostream>

using namespace std;

int main(){

cout << typeid(<Biến Hoặc Giá Trị Hoặc Kiểu Dữ Liệu>).name() << endl;

system("pause");

}

C Standard General Utilities:

1. Cách Import?

#include <cstdlib>

Windows:

1. Cách Import?

#include <windows.h>

1. Tạm Dừng Chương Trình Trong Vài Giây?

Sleep(<Số Mili Giây>)

* <Số Mili Giây> phải là số nguyên dương
* CPU lúc này có thể chuyển sang làm việc khác, hoặc chạy các luồng của chương trình

1. Một Số Kiểu Dữ Liệu Rút Ngắn?

size\_t = unsigned long long

DWORD = usigned long

LONG\_PTR = long long

WINBOOL = int

HANDLE = PVOID = void\*

1. Một Số Giá Trị Đặc Biệt?

INVALID\_HANDLE\_VALUE = FFFFFFFFFFFFFFFF

* Giá trị trên là địa chỉ, kích thước 8 Byte

STANDARD\_RIGHTS\_REQUIRED = F0000 = 983040

SYNCHRONIZE = 10000016 = 1048576

PROCESS\_ALL\_ACCESS = 1F0FFF = 2035711

1. Một Số Hàm Đặc Biệt?

TEXT(<Bất Cứ Thứ Gì>) = <Bất Cứ Thứ Gì>

1. Hủy Cấp Phát Bộ Nhớ?

CloseHandle(<Pointer>);

* Block mà <Pointer> chĩa vào sẽ được tự do nếu không còn bị chĩa vào bởi Handle khác, giống y chang lệnh delete hay free

1. Tạo Handle Đến 1 Process?

<Process Handle> = OpenProcess(2035711, 0, <Process ID>);

* <Process Handle> là Void Pointer
* Ví dụ

void\* foo = OpenProcess(2035711, 0, 790);

1. Ghi Đè 1 Giá Trị Vào Các Ô Nhớ Của 1 Process?

WriteProcessMemory(

<Process Handle>, <Địa Chỉ Chỗ Ghi>, <Địa Chỉ Giá Trị Đè>,

<Số Byte>, 0

);

* <Địa Chỉ Chỗ Ghi> là 1 Pointer mang giá trị là địa chỉ tương đối, hay địa chỉ ảo, tính từ điểm nhìn của Process
* <Địa Chỉ Giá Trị Đè> là 1 Pointer mang giá trị là địa chỉ của ô Byte đầu tiên trong chuỗi <Số Byte> ô Byte liên tiếp sẽ dùng để ghi đè lên các ô Byte ở

<Địa Chỉ Chỗ Ghi>, <Địa Chỉ Chỗ Ghi> = địa chỉ của ô Byte đầu tiên bị ghi đè

1. Đọc Giá Trị Các Ô Nhớ Của 1 Process?

ReadProcessMemory(

<Process Handle>, <Địa Chỉ Chỗ Đọc>, <Địa Chỉ Lưu Giá Trị Đọc>,

<Số Byte>, 0

);

* Ý tưởng tương tự khi ghi đè

Thread – Luồng:

1. Cách Import?

#include <thread>

1. Cách Hoạt Động?

* Khi chương trình bắt đầu chạy, thì 1 Thread ban đầu được sinh ra, Thread này chính là hàm main
* 1 Thread bản chất là 1 hàm đang chạy
* Mỗi Thread khi được tạo ra sẽ được cấp phát 1 Stack riêng, các hàm mà Thread đang dùng tới sẽ đặt trong Stack, CPU sẽ tự động chuyển đổi qua lại giữa các Stack, tạo cảm giác chạy song song
* Các Thread hoàn toàn độc lập, tức là thằng kia tạm dừng thì thằng này vẫn chạy, ví dụ trong hàm main ta tạo 1 Thread mới và tạm dừng main trong vài giây thì Thread mới vẫn chạy và xuất ra CMD

1. Tạo 1 Thread?

std::thread <Tên Thread>(<Function Pointer>, <Các Tham Số Cho Hàm>);

* Ví dụ ta muốn chạy hàm foo với 2 đối số int = 4 và 5 song song với hàm main, thì trong hàm main viết

std::thread bar(foo, 4, 5);

* Khi này 1 Stack mới được tạo ra và khung hàm cho hàm foo được đặt vào Stack này, Stack này chạy song song với Stack chứa hàm main

C Time:

1. Cách Import?

#include <ctime>

1. Số Ngày Các Tháng, Năm Nhuận?

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 31 | 28 | 31 | 30 | 31 | 30 | 31 | 31 | 30 | 31 | 30 | 31 |

* Xét từ năm 0 đến 400, bất cứ năm nào chia cho 4 là năm nhuận trừ 100, 200, 300

1. UTC (Coordinated Universal Time)?

* UTC kí hiệu giờ chuẩn của cả thế giới là giờ của nước anh, tại Greenwich
* Việt Nam nằm tại múi giờ thứ 7, do đó ví dụ giờ UTC đang là 11 giờ sáng ngày thứ 2 thì giờ Việt Nam sẽ là 18 giờ, hay 6 giờ tối ngày thứ 2

1. Trả Về Số Giây Kể Từ Thời Điểm 0 Giờ 0 Phút 0 Giây Ngày 1/1/1970 Tại Greenwich Đến Năm Tháng Ngày Giờ Phút Giây Hiện Tại Cũng Ở Greenwich?

time(<Địa Chỉ Object Lưu Số Giây>);

* <Object Lưu Số Giây> phải thuộc Class long long
* Lệnh trên cũng trả về số giây
* Ví dụ

long long foo, bar;

bar = time(&foo);

* Ta có

|  |  |
| --- | --- |
| foo | 1706356857 |
| bar | 1706356857 |

Vector:

1. Cách Import?

#include <vector>

1. Bản chất?

* Là 1 Pointer từ Stack đến mảng trong Heap
* Nó sẽ có 3 mốc, mốc dưới cùng A là vị trí đầu mảng, mốc ở giữa B là vị trí phần tử cuối cùng, mốc trên cùng C là vị trí kết thúc mảng, khoảng giữa A và C là Block
* Khi thay đổi 1 phần tử bất kì thông qua toán tử Subscript, trong Block hoặc ngoài Block, thì các mốc không thay đổi
* Khi đẩy 1 phần tử vào cuối mảng, thì sẽ chèn ở vị trí kế sau B, nếu vị trí này vượt qua C, thì cấp phát lại bộ nhớ như lệnh realloc, Copy toàn bộ phần tử của mảng vào mảng mới, mảng này có kích thước Block gấp đôi mảng cũ, các phần tử chưa được xác định giá trị thì giá trị ngẫu nhiên, A, B, C cũng cập nhật, B thì sẽ nằm ở vị trí phần tử mới chèn vào
* Khi lặp qua Vector dùng lệnh For, thì nó sẽ lặp từ A tới B
* Khi chưa khởi tạo giá trị ban đầu, kích thước Block = 0, khi chèn phần tử đầu tiên, kích thước Block = 1, nếu có khởi tạo giá trị ban đầu, kích thước Block = số phần tử của mảng khởi tạo

1. Tạo 1 Vector?

std::vector<<Class>> <Vector> = <Mảng>;

* Ví dụ

std::vector<int> foo = {4, 5, 6, 7};

1. Đẩy 1 Phần Tử Vào Cuối Vector?

<Vector>.push\_back(<Giá Trị Phần Tử>);

Map:

1. Cách Import?

#include <map>

1. Bản Chất?

* Là 1 Pointer từ cấu trúc dữ liệu Red Black Tree trong Heap, với giá trị Node dùng để tìm kiếm có thể là mảng Char, số, …, các giá trị này gọi là Key, thứ ta muốn lấy ra từ các Node là Value, các Node sẽ lưu 1 cách rời rạc

1. Tạo 1 Map?

std::map<<Key Class>, <Value Class>> <Map>;

* Ví dụ

std::map<std::string, int> foo;

* Ta có thể khởi tạo Map với các cặp Key Value ban đầu
* Ví dụ

std::map<std::string, int> foo = {{"bar", 4}, {"alice", 8}};

* Để truy cập Value của 1 Key trong Map

<Map>[<Key>]

* Ví dụ

foo["bar"]

* Để thêm 1 cặp Key Value mới vào Map

<Map>[<Key Mới>] = <Giá Trị>;

* Nếu <Key Mới> đã tồn tại thì ghi đè
* Trả về số cặp Key Value trong Map

<Map>.size()

* Xóa 1 cặp Key Value ra khỏi Map

<Map>.erase(<Key>);

Tool Help – Công Cụ Trợ Giúp:

1. Cách Import?

#include <tlhelp32.h>

1. Một Số Giá Trị Đặc Biệt?

TH32CS\_SNAPPROCESS = 2

1. Trả Về Snap Shot Toàn Bộ Process Trong RAM Thời Điểm Hiện Tại?

void\* <Snap Shot> = CreateToolhelp32Snapshot(2, 0);

* Ảnh của toàn bộ Process trong RAM sẽ được chụp, sau đó ảnh này sẽ được <Snap Shot> chĩa vào, do đó sau khi sử dụng xong <Snap Shot> thì nên đóng nó lại = CloseHandle hoặc để chương trình tự động khi kết thúc, tránh để tài nguyên ứ đọng
* <Snap Shot> có 1 mốc, vị trí ban đầu nằm ngay trước Process đầu tiên

1. Tạo Process Entry Dùng Để Chứa 1 Process Nào Đó?

PROCESSENTRY <Process Entry>;

1. Đưa Mốc Của Snap Shot Tới Process Kế Tiếp Và Gán Process Đó Cho Process Entry?

Process32Next(<Snap Shot>, &<Process Entry>);

1. Trả Về Process Đầu Tiên Của Snap Shot Đéo Quan Tâm Mốc?

Process32First(<Snap Shot>, &<Process Entry>);

1. Trả Về Thông Tin Của Process Chứa Trong Process Entry?

<Tên> = <Process Entry>.szExeFile;

<ID> = <Process Entry>.th32ProcessID;

* <Tên> là 1 String, ví dụ "foo.exe"
* <ID> là 1 số nguyên

String – Chuỗi:

1. Cách Import?

#include <string.h>

File Stream:

1. Cách Import?

#include <fstream>

1. Cách Hoạt Động?

* Y chang cách hoạt động của File trong Python
* Ở kế ngay sau kí tự cuối cùng của 1 File, sẽ có 1 chuỗi Bit đặc biệt, khác nhau theo hệ thống, đại diện cho EOF (End Of File)

1. Mở 1 File Ở Chế Độ Ghi Và Gán Nó Vào 1 File Object?

std::ofstream <File Object>(<Đường Dẫn Đến File Có Phần Mở Rộng>);

* Hoặc

std::ofstream <File Object>;

<File Object>.open(<Đường Dẫn Đến File Có Phần Mở Rộng>);

* Ví dụ

std::ofstream foo("bar.txt");

* Lệnh sau y chang lệnh write trong Python

<File Object> << <Nội Dung>;

* Về bản chất, cout và <File Object> tương tự nhau
* Ví dụ

foo << "thang cho ma";

1. Mở 1 File Ở Chế Độ Đọc Và Gán Nó Vào 1 File Object?

std::ifstream <File Object>(<Đường Dẫn Đến File Có Phần Mở Rộng>);

* Trả về chuỗi kí tự của File vào 1 String bắt đầu từ vị trí hiện tại của con trỏ, đến hết kí tự đứng trước khoảng trắng đầu tiên, đồng thời đưa con trỏ tới vị trí kế ngay sau kí tự cuối cùng của chuỗi kí tự, rồi dịch sang phải cho đến khi kí tự chĩa vào không phải dấu cách

<File Object> >> <Biến Lưu Trữ>;

* Về bản chất, cin và <File Object> tương tự nhau
* Ví dụ

foo >> bar;

* Trả về kí tự ở vị trí con trỏ, và dịch phải con trỏ 1 kí tự

<File Object>.get()

1. Đóng 1 File Object?

<File Object>.close();

Algorithm – Thuật Toán:

1. Cách Import?

#include <algorithm>

1. Hoán Vị Mảng?

std::next\_permutation(<Pointer 1>, <Pointer 2>);

* <Pointer 1> và <Pointer 2> phải cùng Class
* Ban đầu cô lập vùng nhớ, bắt đầu từ địa chỉ <Pointer 1> chĩa vào, cho tới địa chỉ ngay trước địa chỉ <Pointer 2> chĩa vào, sau đó chia vùng nhớ này thành các ô, tùy theo Class của 2 Pointer, ví dụ nếu là int\* thì chia 4 ô Byte một, mỗi 4 ô Byte một tạm gọi là 1 khối
* Sau đó sẽ hoán vị các khối sao cho ra được dãy nhị phân nhỏ nhất và lớn hơn dãy nhị phân hiện tại từ MSB đến LSB, MSB là khối ở vị trí <Pointer 1>, LSB là khối ngay trước <Pointer 2>, ví dụ [4, 9, 8] thì dãy nhỏ nhất và lớn hơn dãy này là [8, 4, 9], vâng, 498 < 849
* Nếu có thể hoán vị để cho ra dãy tiếp theo lớn hơn, thì lệnh trên sẽ trả về True, nếu không, thì nó sẽ trả về False, đồng thời hoán vị thành dãy nhỏ nhất có thể, ví dụ từ [9, 8, 7] thành [7, 8, 9]
* Ví dụ

int a[4] = {7, 9, 4, 3};

bool foo = std::next\_permutation(a, a + 4);

* Khi này mảng a sẽ là [9, 3, 4, 7], và foo = True