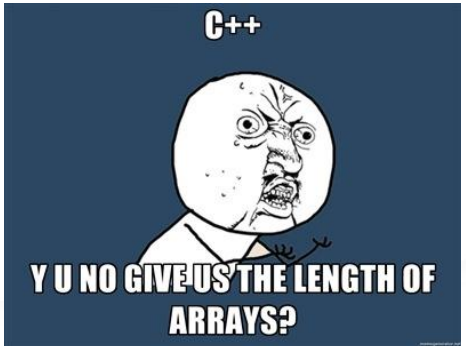
National Cheng Kung University

COLLEGE of
ELECTRICAL ENGINEERING
and COMPUTER SCIENCE

Lecture 4 -
Array and Vector

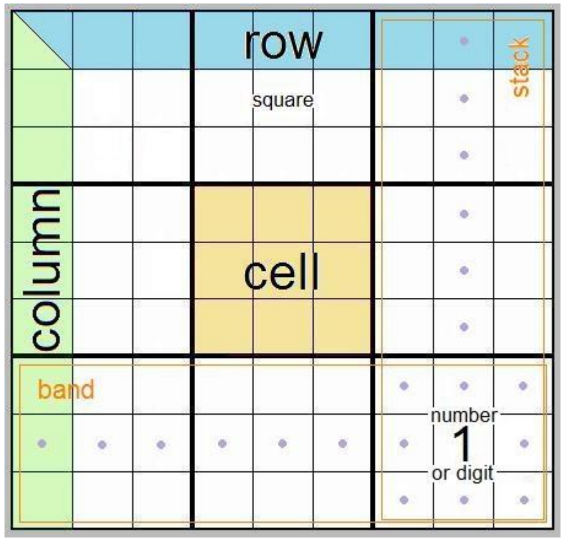
Yean-Ru Chen
chenyr@mail.ncku.edu.tw

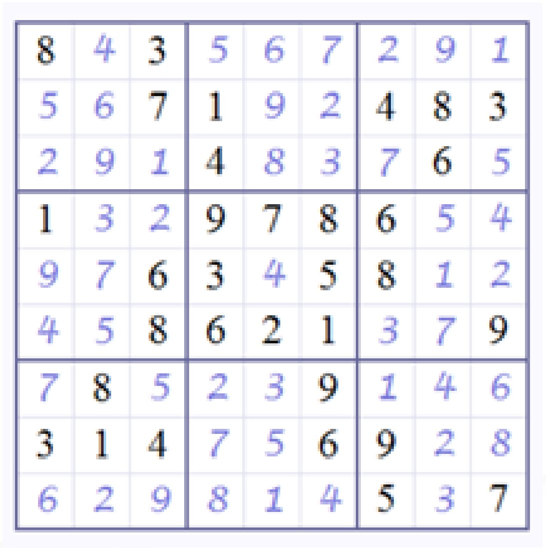
Contents ack. to Prof. M.-H. Tsai @ CSIE NCKU



Sudoku Validator

- A Sudoku validator reads a Sudoku answer from a file, and then checks if the answer is valid or not.
- A Sudoku answer is a 9×9 grid filled with digits so that **each column**, **each row**, and **each of the nine 3×3 sub-grids (called cells)** that compose the grid contains all of the digits from 1 to 9.



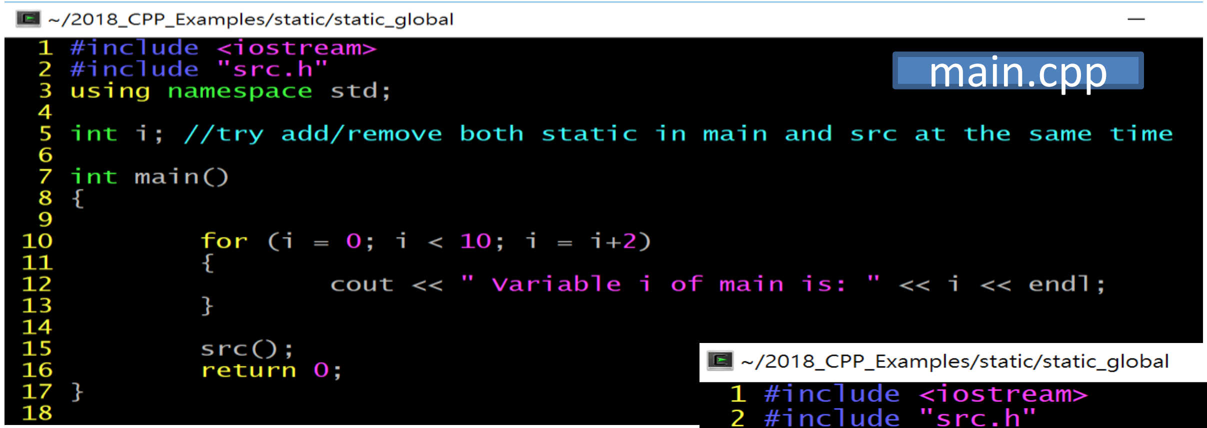


8	4	3	5	6	7	2	9	1
5	6	7	1	9	2	4	8	3
2	9	1	4	8	3	7	6	5
1	3	2	9	7	8	6	5	4
9	7	6	3	4	5	8	1	2
4	5	8	6	2	1	3	7	9
7	8	5	2	3	9	1	4	6
3	1	4	7	5	6	9	2	8
6	2	9	8	1	4	5	3	7

Sudoku.h

```
#include <iostream>
class Sudoku {
public:
    Sudoku();
    Sudoku(const int init_map[]);
    void setMap(const int  set_map[]);
    int getElement(int index);
    bool isCorrect();
    static const int sudokuSize = 81;

private:
    bool checkUnity(int arr[]);
    int map[sudokuSize];
};
```

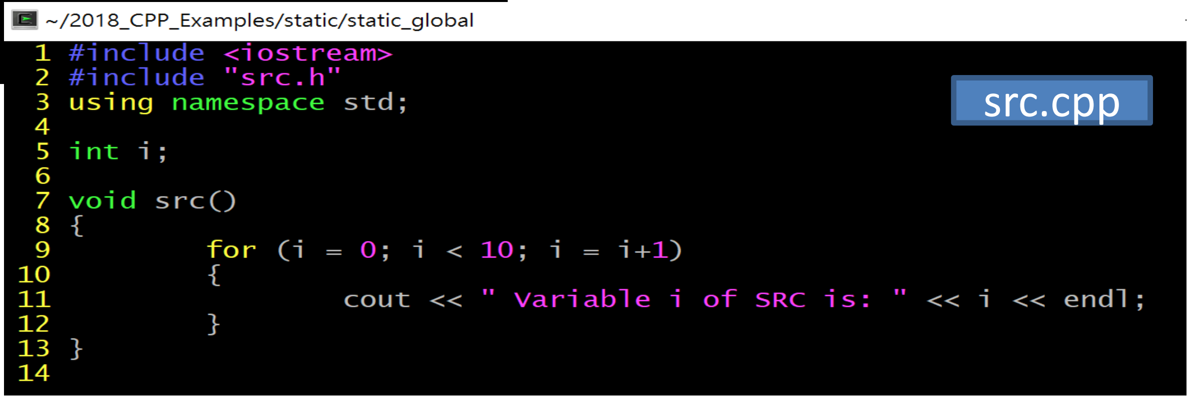
Not use static for global variables in two functions



~/2018_CPP_Examples/static/static_global

main.cpp

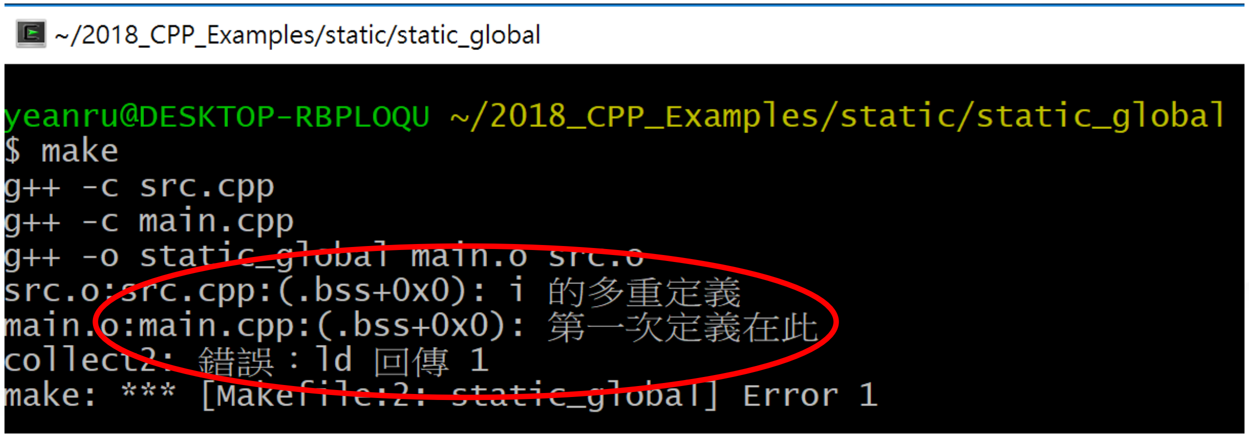
```cpp
#include <iostream>
#include "src.h"
using namespace std;

int i; //try add/remove both static in main and src at the same time

int main()
{

        for (i = 0; i < 10; i = i+2)
        {
                cout << " Variable i of main is: " << i << endl;
        }

        src();
        return 0;
}

```



~/2018_CPP_Examples/static/static_global

src.cpp

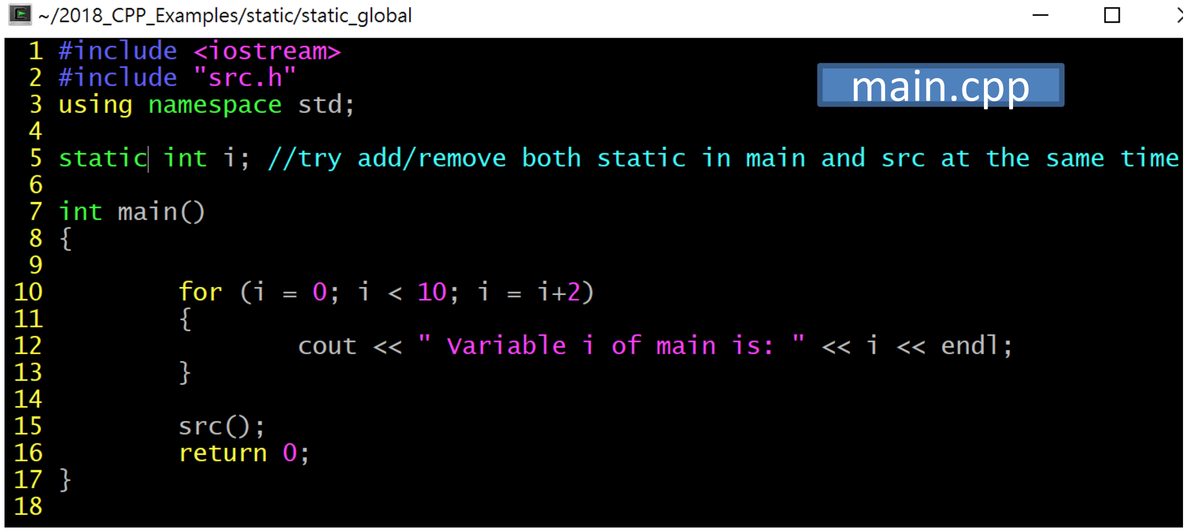
```cpp
#include <iostream>
#include "src.h"
using namespace std;

int i;

void src()
{
        for (i = 0; i < 10; i = i+1)
        {
                cout << " Variable i of SRC is: " << i << endl;
        }
}

```



~/2018_CPP_Examples/static/static_global

```
yeanru@DESKTOP-RBPLOQU ~/2018_CPP_Examples/static/static_global
$ make
g++ -c src.cpp
g++ -c main.cpp
g++ -o static_global main.o src.o
src.o:src.cpp:(.bss+0x0): i 的多重定義
main.o:main.cpp:(.bss+0x0): 第一次定義在此
collect2: 錯誤：ld 回傳 1
make: *** [Makefile:2: static_global] Error 1
```

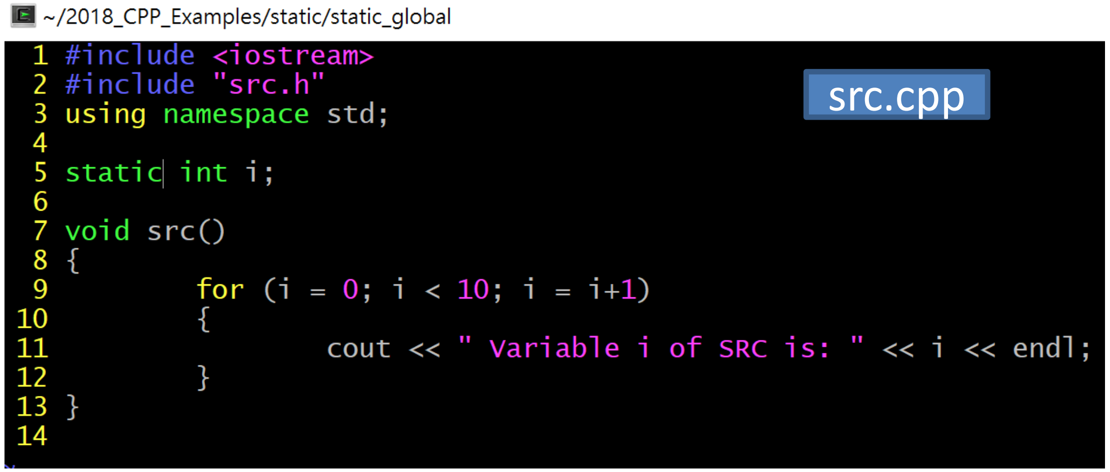
Use static for global variables in two functions



~/2018_CPP_Examples/static/static_global

main.cpp

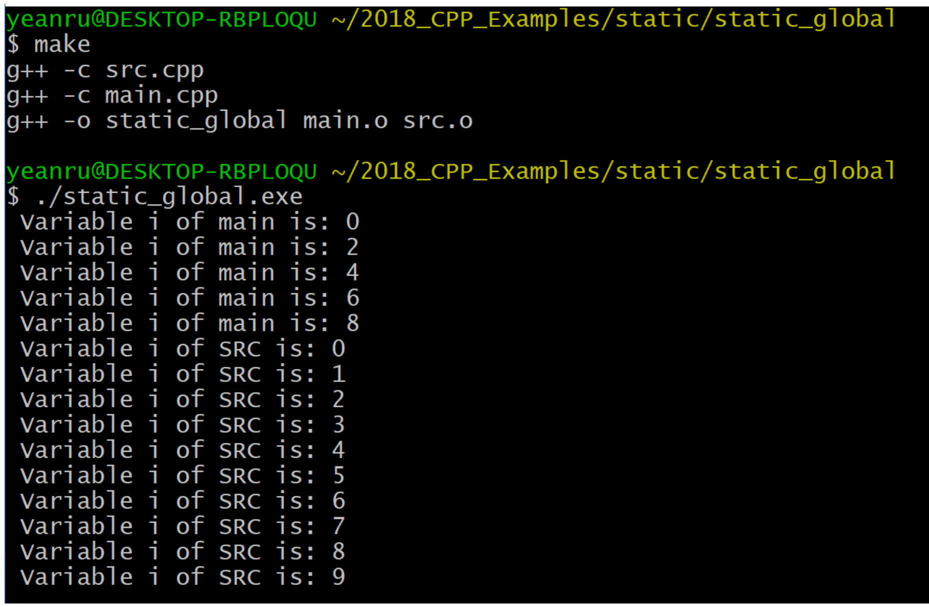
```cpp
#include <iostream>
#include "src.h"
using namespace std;

static int i; //try add/remove both static in main and src at the same time

int main()
{

	for (i = 0; i < 10; i = i+2)
	{
		cout << " Variable i of main is: " << i << endl;
	}

	src();
	return 0;
}

```



~/2018_CPP_Examples/static/static_global

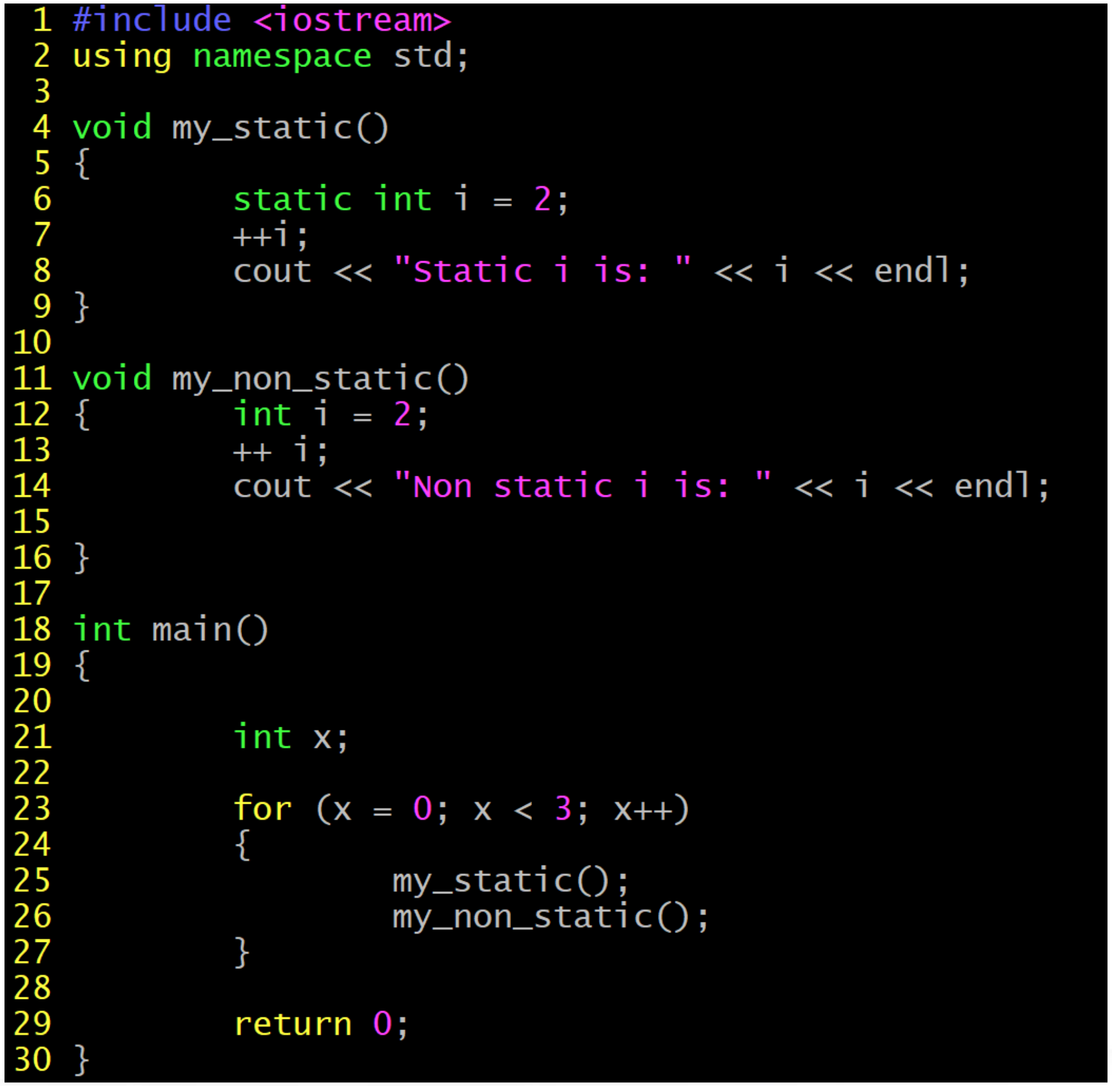
src.cpp

```cpp
#include <iostream>
#include "src.h"
using namespace std;

static int i;

void src()
{
	for (i = 0; i < 10; i = i+1)
	{
		cout << " Variable i of SRC is: " << i << endl;
	}
}

```

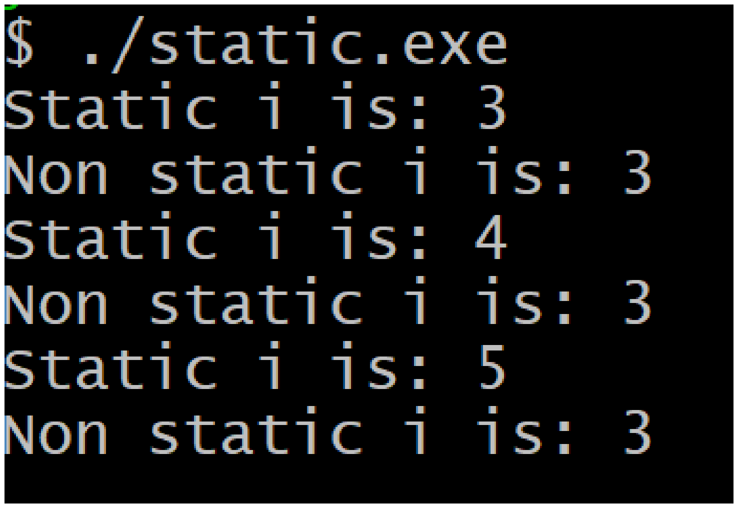


```
yeanru@DESKTOP-RBPLOQU ~/2018_CPP_Examples/static/static_global
$ make
g++ -c src.cpp
g++ -c main.cpp
g++ -o static_global main.o src.o

yeanru@DESKTOP-RBPLOQU ~/2018_CPP_Examples/static/static_global
$ ./static_global.exe
 Variable i of main is: 0
 Variable i of main is: 2
 Variable i of main is: 4
 Variable i of main is: 6
 Variable i of main is: 8
 Variable i of SRC is: 0
 Variable i of SRC is: 1
 Variable i of SRC is: 2
 Variable i of SRC is: 3
 Variable i of SRC is: 4
 Variable i of SRC is: 5
 Variable i of SRC is: 6
 Variable i of SRC is: 7
 Variable i of SRC is: 8
 Variable i of SRC is: 9
```

Static local variable



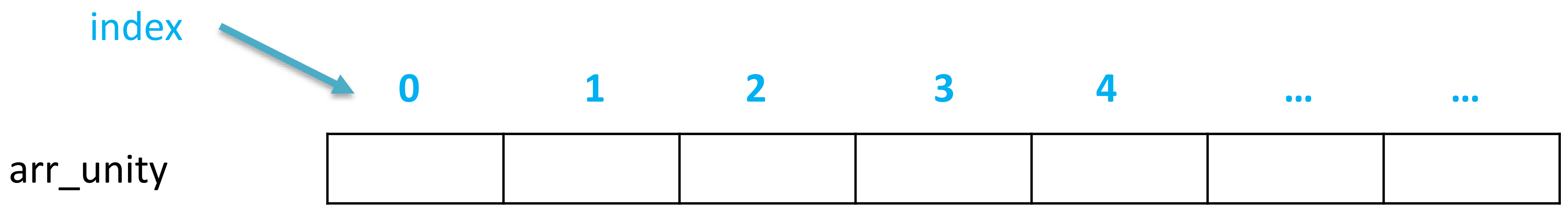
```cpp
#include <iostream>
using namespace std;

void my_static()
{
        static int i = 2;
        ++i;
        cout << "Static i is: " << i << endl;
}

void my_non_static()
{       int i = 2;
        ++ i;
        cout << "Non static i is: " << i << endl;

}

int main()
{

        int x;

        for (x = 0; x < 3; x++)
        {
                my_static();
                my_non_static();
        }

        return 0;
}
```



```
$ ./static.exe
Static i is: 3
Non static i is: 3
Static i is: 4
Non static i is: 3
Static i is: 5
Non static i is: 3
```

Sudoku.cpp

```cpp
#include "Sudoku.h"
using namespace std;

Sudoku::Sudoku()
{
    for(int i=0; i<sudokuSize; ++i)
        map[i] = 0;
}
Sudoku::Sudoku(const int init_map[])
{
    for(int i=0; i<sudokuSize; ++i)
        map[i] = init_map[i];
}

void Sudoku::setMap(const int set_map[])
{
    for(int i=0; i<sudokuSize; ++i)
        map[i] = set_map[i];
}
int Sudoku::getElement(int index)
{
    return map[index];
}

bool Sudoku::checkUnity(int arr[])
{
    int arr_unity[9];   // counters

    for(int i=0; i<9; ++i)
        arr_unity[i] = 0;    // initialize
    for(int i=0; i<9; ++i)
        ++arr_unity[arr[i]-1];    // count
    for(int i=0; i<9; ++i)
        if(arr_unity[i] != 1)   // all element
            return false;        // must be 1
    return true;
}

```



若 cell 裡是: 123456789 (這是 arr [i] 的內容)
則 arr_unity 裡存的內容是: 111111111 (從 index 0 -8 內全部存1)

若 cell 裡是: 112345678 (這是 arr [i] 的內容)
則 arr_unity 裡存的內容是: 211111110 (從 index 0-8 內存的值)

++i vs. i++

- int main()
```
{
  int i = 1;
  cout << “Step1: “ << i++ << endl;
  cout << “Step2: “ << i << endl;
  return 0;
  }
```

```
Step1: 1
Step2: 2
```

- int main()
```
{
  int i = 1;
  cout << “Step3: “ << i << endl;
  cout << “Step4: “ << ++i << endl;
  cout << “Step5: “ << i << endl;
  return 0;
  }
```

```
Step3: 1
Step4: 2
Step5: 2
```

Sudoku.cpp (cont.)

```
bool Sudoku::isCorrect()
{
    bool check_result;
    int check_arr[9];
    int location;
    for(int i=0; i<81; i+=9)     // check rows
    {
        for(int j=0; j<9; ++j)
            check_arr[j] = map[i+j];
        check_result = checkUnity(check_arr);
        if(check_result == false)
            return false;
    }
    for(int i=0; i<9; ++i)    // check columns
    {
        for(int j=0; j<9; ++j)
            check_arr[j] = map[i+9*j];
        check_result = checkUnity(check_arr);
        if(check_result == false)
            return false;
    }
    for(int i=0; i<9; ++i)   // check cells
    {
        for(int j=0; j<9; ++j)
        {
            location = 27*(i/3) + 3*(i%3)
                       +9*(j/3) + (j%3);
            check_arr[j] = map[location];
        }
        check_result =
                checkUnity(check_arr);
        if(check_result == false)
            return false;
    }
    return true;
}
```

public static const Data Member

- Note that the size of the array is specified as a public static const data member.

  (1) public so that it's accessible to the clients of the class.

  (2) const so that this data member is constant.

  (3) static so that the data member is shared by all objects of the class

- static data members are also known as class variables.

- When objects of a class containing static data members are created, all the objects share one copy of the class's static data members.

Error: Initialization of *const* Data Member

```
> cat -n const1.cpp
    1  class Cls {
    2  public:
    3          Cls(){ x = 3;}
    4          const int x;
    5  };
    6  int main() { return 0; }
> g++ -o const1 const1.cpp
const1.cpp: In constructor `Cls::Cls()':
const1.cpp:3: error: uninitialized member `Cls::x' with `const' type `const int'
const1.cpp:3: error: assignment of read-only data-member `Cls::x'
```

Initialization of *const* Data Member (cont.)

```
> cat -n const2.cpp
     1      class Cls {
     2      public:   const int x = 3;
     3      };
     4      int main() { return 0; }
> g++ -o const2 const2.cpp
const2.cpp:2:23: warning: in-class initialization of non-static data member is a
      C++11 extension [-Wc++11-extensions]
public:   const int x = 3;
                      ^
1 warning generated.
```

```
> cat -n const3.cpp
     1      class Cls {
     2      public:   Cls():x(3) {}
     3                const int x;
     4      };
     5      int main()  {  return 0; }
> g++ -o const3 const3.cpp
>
```

Initialization of *static const* Data Member

static_const1.cpp

```
1  class Cls {
2  public:   Cls():x(3) {}
3            static const int x;
4  };
5  int main()  {  return 0;  }
```

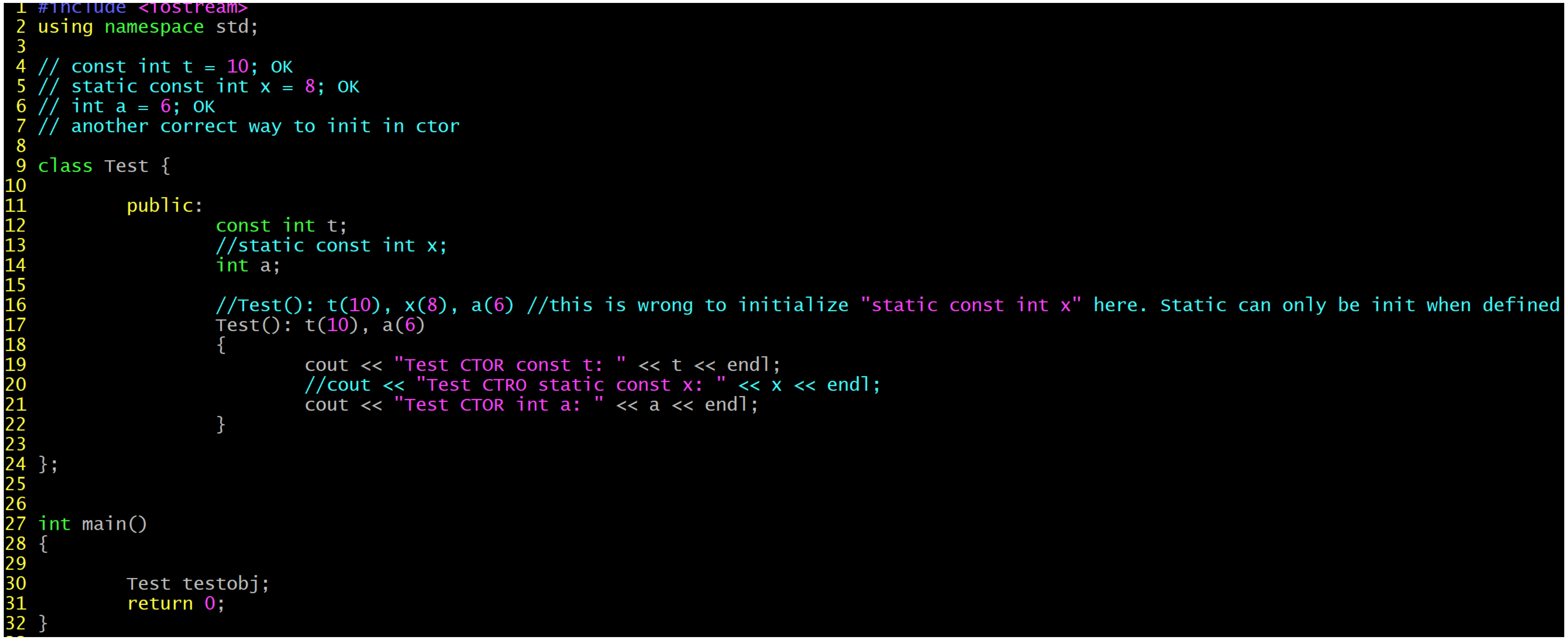
static_const2.cpp

```
1  class Cls {
2  public:   Cls(){}
3        static const int x = 3;
4  };
5  int main()  {  return 0;  }
```

```
> g++ -o static_const1 static_const1.cpp
static_const1.cpp: In constructor `Cls::Cls()':
static_const1.cpp:2: error: `const int Cls::x' is a static data member; it can only be initialized at its definition
```

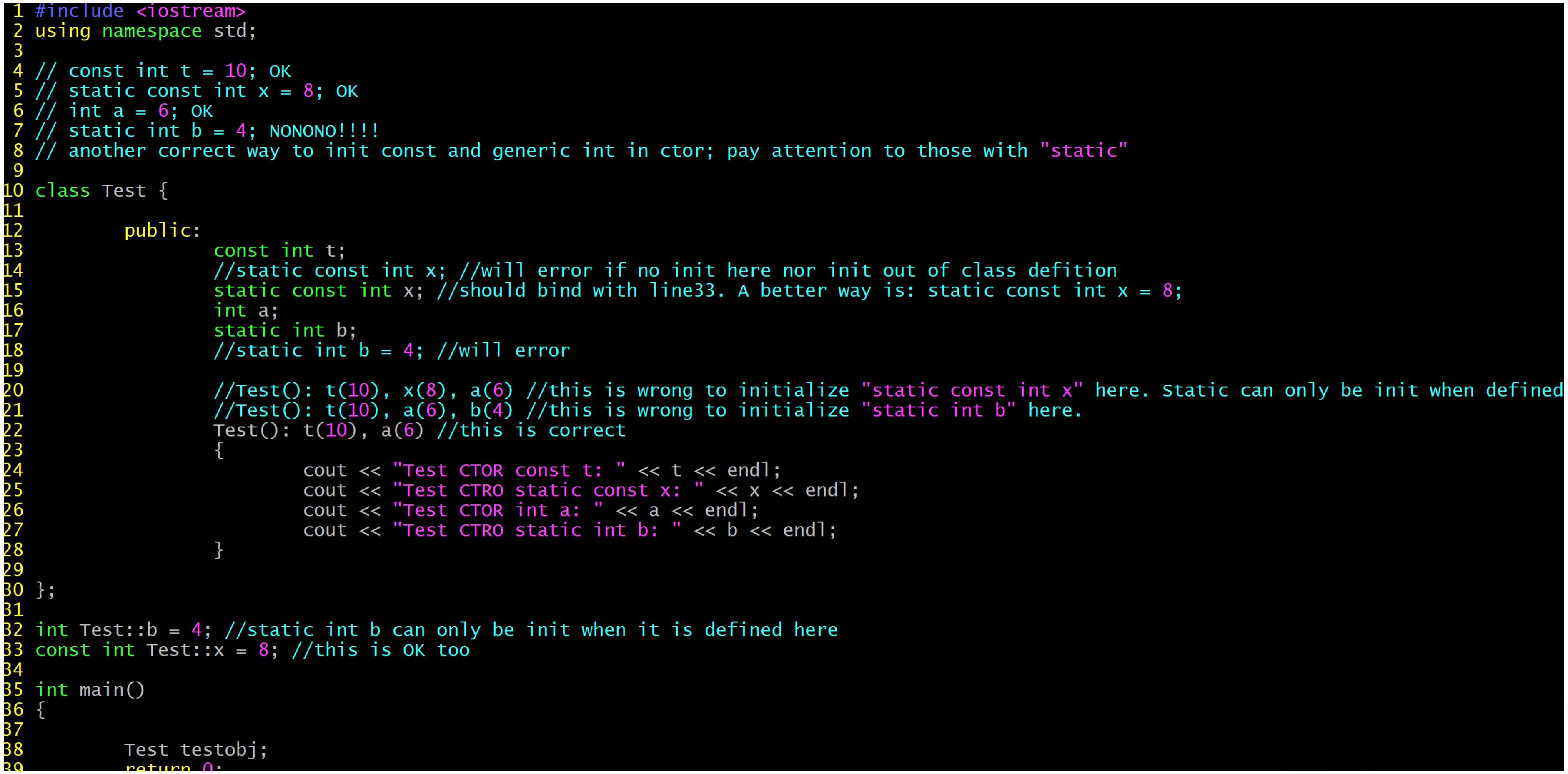
```
> g++ -o static_const2 static_const2.cpp
>
```

2018 g++: initialize variable in class



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

// const int t = 10; OK
// static const int x = 8; OK
// int a = 6; OK
// another correct way to init in ctor

class Test {

        public:
                const int t;
                //static const int x;
                int a;

                //Test(): t(10), x(8), a(6) //this is wrong to initialize "static const int x" here. Static can only be init when defined
                Test(): t(10), a(6)
                {
                        cout << "Test CTOR const t: " << t << endl;
                        //cout << "Test CTRO static const x: " << x << endl;
                        cout << "Test CTOR int a: " << a << endl;
                }

};


int main()
{

        Test testobj;
        return 0;
}
```

2018 g++: initialize variable in class (cont.)



```
1  #include <iostream>
2  using namespace std;
3
4  // const int t = 10; OK
5  // static const int x = 8; OK
6  // int a = 6; OK
7  // static int b = 4; NONONO!!!!
8  // another correct way to init const and generic int in ctor; pay attention to those with "static"
9
10 class Test {
11
12         public:
13                 const int t;
14                 //static const int x; //will error if no init here nor init out of class defition
15                 static const int x; //should bind with line33. A better way is: static const int x = 8;
16                 int a;
17                 static int b;
18                 //static int b = 4; //will error
19
20                 //Test(): t(10), x(8), a(6) //this is wrong to initialize "static const int x" here. Static can only be init when defined
21                 //Test(): t(10), a(6), b(4) //this is wrong to initialize "static int b" here.
22                 Test(): t(10), a(6) //this is correct
23                 {
24                         cout << "Test CTOR const t: " << t << endl;
25                         cout << "Test CTRO static const x: " << x << endl;
26                         cout << "Test CTOR int a: " << a << endl;
27                         cout << "Test CTRO static int b: " << b << endl;
28                 }
29
30 };
31
32 int Test::b = 4; //static int b can only be init when it is defined here
33 const int Test::x = 8; //this is OK too
34
35 int main()
36 {
37
38         Test testobj;
39         return 0;
```

Size of Object with *static const* and *const* Data Members

```
#include <iostream>
using namespace std;
class Cls {
public:   Cls():y(4){}
                static const int x = 3;
                const int y;
};
int main()
{
          Cls obj;
          cout << "sizeof(Cls) = " << sizeof(Cls) << endl;
          cout << "sizeof(obj) = " << sizeof(obj) << endl;
          return 0;
}
```

```
Output:
sizeof(Cls) = 4
sizeof(obj) = 4
```

static Data Member

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

class Cls {
public:      Cls(){  NumObject++;  }
    static int NumObject;
};
int Cls::NumObject = 0;
int main()
{
   cout << Cls::NumObject << endl;
   Cls obj1;
   cout << Cls::NumObject << endl;
   Cls obj2;
   cout << obj1.NumObject << endl;
   cout << obj2.NumObject << endl;
   return 0;
}
```

Just Declaration

Definition (Do not use “static” here.)

```
Output:
0
1
2
2
```

static Data Member (cont.)

- A static data member can be accessed within the class definition and the member-function definitions like any other data member.
- A public static data member can also be accessed outside of the class, even when no objects of the class exist, using the class name followed by the binary scope resolution operator (::) and the name of the data member.

Sample Input and Sample Output

```
> cat su_infile
2
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
8 6 5 3 2 9 4 1 7
2 4 3 1 7 5 8 6 9
1 9 7 6 8 4 5 2 3
3 1 9 2 5 8 6 7 4
4 2 6 7 9 1 3 5 8
5 7 8 4 3 6 1 9 2
7 5 4 9 1 3 2 8 6
6 8 2 5 4 7 9 3 1
9 3 1 8 6 2 7 4 5
```

Number of cases

Map of case #1

Map of case #2

```
> ./sudoku_validate
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
INCORRECT
8 6 5 3 2 9 4 1 7
2 4 3 1 7 5 8 6 9
1 9 7 6 8 4 5 2 3
3 1 9 2 5 8 6 7 4
4 2 6 7 9 1 3 5 8
5 7 8 4 3 6 1 9 2
7 5 4 9 1 3 2 8 6
6 8 2 5 4 7 9 3 1
9 3 1 8 6 2 7 4 5
CORRECT
```

Validation Result

sudoku_validate.cpp

```cpp
#include <cstdlib>
#include <iostream>
#include <fstream>
#include "Sudoku.h"
#define MAX_CASE    100
using namespace std;
int main()
{
    int sudoku_in[Sudoku::sudokuSize];
    Sudoku su[MAX_CASE];
    ifstream in("su_infile",ios::in);
    int num_case;
    in >> num_case;
    for(int j=0; j<num_case; ++j)
    {
        for(int i=0; i<Sudoku::sudokuSize; ++i)
            in >> sudoku_in[i];      // read in map
        su[j].setMap(sudoku_in);  // set map
    }
    for(int j=0; j<num_case; ++j)
    {       // print out the maps
        for(int i=0; i<Sudoku::sudokuSize; ++i)
        {
            cout << su[j].getElement(i) << " ";
            if(i % 9 == 8 )
                cout << endl;
        }
        if(su[j].isCorrect()) // validation results
            cout << "CORRECT\n";
        else
            cout << "INCORRECT\n";
    }
    return 0;
}
```

Replacing Array with *vector*

```
#include <vector>
#include <cstdlib>
#include <iostream>
#include <fstream>
#include "Sudoku.h"
using namespace std;
int main()
{
    int sudoku_in[Sudoku::sudokuSize];
    Sudoku su_tmp;
    vector<Sudoku> su;
    ifstream in("su_infile",ios::in);
   int num_element, num_case;
   in >> num_case;
// num_case is not used in this program
    cout << "size = " <<
                su.size() << endl;
    num_element = 0;
    while(in >> sudoku_in[num_element++])
    {                   // read in map
       if(num_element >=
                  Sudoku::sudokuSize) {
          su_tmp.setMap(sudoku_in);
          num_element = 0;
          su.push_back(su_tmp);
       }
    }
    cout << "size = " << su.size() << endl;
    cout << su[0].isCorrect() << endl;
    for(int i = 1; i<su.size(); ++i)
       cout << su.at(i).isCorrect() << endl;

    return 0;
}
```

```
> ./sudoku_validate2
size = 0
size = 2
0
1
```

C++ Standard Library Class Template vector

- C-style pointer-based arrays have great potential for errors and are not flexible
- A program can easily "walk off" either end of an array, because C++ does not check whether subscripts fall outside the range of an array.

```
arr[-1]
```

- Two arrays cannot be meaningfully compared with equality operators or relational operators.

```
if(arr1 == arr2)
```

- When an array is passed to a general-purpose function designed to handle arrays of any size, the size of the array must be passed as an additional argument.

```
func(arr, size)
```

- One array cannot be assigned to another with the assignment operator(s).

```
arr1 = arr2
```

C++ Standard Library Class Template vector (cont.)

- C++ Standard Library class template vector represents a more robust type of array featuring many additional capabilities.
- Standard class template vector is defined in header <vector> and belongs to namespace std.
- By default, all the elements of a vector object are set to 0.
- vectors can be defined to store any data type.

```
vector<int> v1;
vector<Sudoku> v2;
```

- vector member function size obtain the number of elements in the vector.

```
cout << v.size();
```

- vector objects can be compared with one another using the equality operators.

```
if(v1 == v2)
```

C++ Standard Library Class Template vector (cont.)

- You can create a new vector object that is initialized with the contents of an existing vector by using its copy constructor.

```
vector<Sudoku> v2(v1);
```

- You can use the assignment (=) operator with vector objects.

```
v1 = v2;
```

- You can use square brackets, [], to access the elements in a vector. As with C-style pointer-based arrays, C++ does not perform any bounds checking when vector elements are accessed with square brackets.

```
v[1];
```

- Standard class template vector provides bounds checking in its member function at, which "throws an exception" if its argument is an invalid subscript.

```
v.at(1);
```

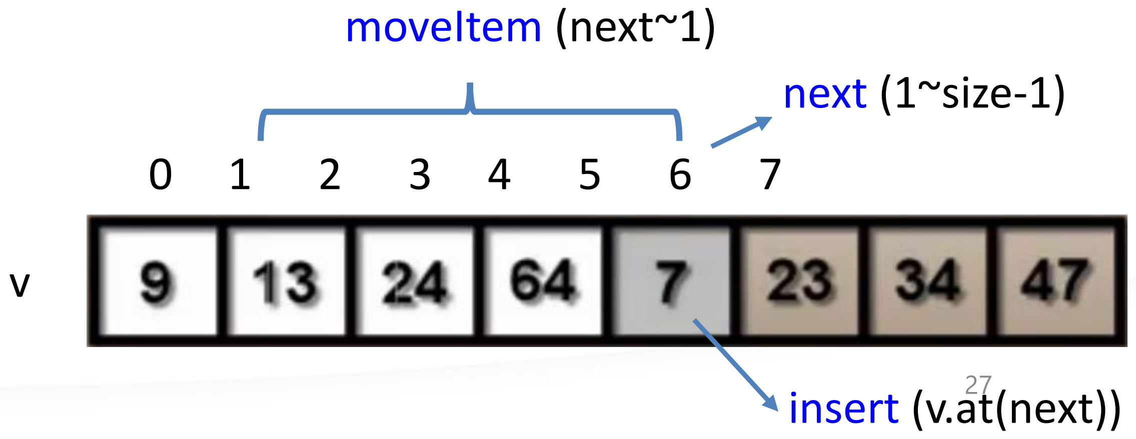
Sorting a Vector with Insertion Sort

```cpp
#include <vector>
#include <iomanip>
#include <iostream>
using namespace std;

int main()
{
    const int size = 8;
    int init_array[size] =
          {64, 24, 13, 9, 7, 23, 34, 47};
    vector<int> v(size);
    int insert, moveItem;

    cout << "Unsorted array:\n";
    for(int i=0; i<size; ++i)
    {
        v.at(i) = init_array[i];
        cout << setw(4) << v.at(i);
    }
    cout << endl;

    cout << "Step-by-step:\n";
    for(int next=1;next<size;++next)
    {
        insert = v.at(next);
        moveItem = next;
        while((moveItem>0) &&
                (v.at(moveItem-1) > insert))
        {
            v.at(moveItem) = v.at(moveItem-1);
            --moveItem;
        }
        v.at(moveItem) = insert;
        for(int i=0; i<size; ++i)
            cout << setw(4) << v.at(i);
        cout << endl;
    }

    return 0;
}
```

Sorting a Vector with Insertion Sort (cont.)

```
for(int next=1;next<size;++next)
{
    insert = v.at(next);
    moveItem = next;
    while((moveItem>0) &&
             (v.at(moveItem-1) > insert))
    {
        v.at(moveItem) = v.at(moveItem-1);
        --moveItem;
    }
    v.at(moveItem) = insert;
...
}
```

```
Output:
Unsorted array:
 64  24  13   9   7  23  34  47
Step-by-step:
 24  64  13   9   7  23  34  47
 13  24  64   9   7  23  34  47
  9  13  24  64   7  23  34  47
  7   9  13  24  64  23  34  47
  7   9  13  23  24  64  34  47
  7   9  13  23  24  34  64  47
  7   9  13  23  24  34  47  64
```



Bubble Sort ($O(n^2)$)

- https://www.youtube.com/watch?v=lyZQPjUT5B4

Quick Sort (O(n log n))

- https://www.youtube.com/watch?v=ywWBy6J5gz8

Using *sort()* in C++ Standard Library

```cpp
#include <vector>
#include <algorithm>
#include <iostream>
#include <cstdlib>
#include "Clock.h"
using namespace std;
void insertion_sort(vector<int> & v)
{
    int insert, moveItem;
    for(int next=1;next<v.size();++next)
    {
        insert = v.at(next);
        moveItem = next;
        while((moveItem>0) &&
              (v.at(moveItem-1) > insert))
        {
            v.at(moveItem) = v.at(moveItem-1);
            --moveItem;
        }
        v.at(moveItem) = insert;
    }
}
```

```cpp
int main()
{
    Clock clk;
    const int size = 100000;
    vector<int> v1(size),v2;
    srandom(time(NULL));
    for(int i=0; i<size; ++i)
        v1.at(i) = random();
    v2 = v1;    clk.start();
    sort(v1.begin(), v1.end());
    cout << "sort(): " <<
     clk.getElapsedTime() << " seconds\n";
    cout << "v1 and v2 are "<<
     ((v1==v2)?"the same.\n":"different.\n");
    clk.start();
    insertion_sort(v2);
    cout << "insertion_sort(): " <<
     clk.getElapsedTime() << " seconds\n";
    cout << "v1 and v2 are "<<
     ((v1==v2)?"the same.\n":"different.\n");
    return 0;
}
```

```
sort(): 0.0547
seconds
v1 and v2 are
different.
insertion_sort():
154.26 seconds
v1 and v2 are the
same.
```

Clock.h and Clock.cpp

Clock.h

```cpp
#include <ctime>
using namespace std;
class Clock {
    public:
        Clock();
        Clock(clock_t s);
        void start();
        void setStart(clock_t start_ts);
        clock_t getStart();
        double getElapsedTime();
    private:
        clock_t start_ts;
};
```

Clock.cpp

```cpp
#include "Clock.h"
Clock::Clock() { setStart(0); }
Clock::Clock(clock_t s) {
    setStart(s);
}
void Clock::start() {
    setStart(clock());
}
void Clock::setStart(clock_t ts) {
    start_ts = (ts>0)?ts:clock();
}
clock_t Clock::getStart() {
    return start_ts;
}
double Clock::getElapsedTime() {
    return static_cast<double>(clock()-getStart())
                    / CLOCKS_PER_SEC ;
}
```

Reference

- Insertion Sort Concept, http://www.youtube.com/watch?v=Fr0SmtN0IJM&t=126
- Insertion Sort Example, http://www.youtube.com/watch?v=c4BRHC7kTaQ&t=75
- Insertion Sort with Romanian Folk Dance, http://www.youtube.com/watch?v=ROalU379l3U