LAB 4

CLASS -1

Department of Electronics Engineering

National Chiao Tung University

Class

- C++ supports object-oriented programming (OOP)
- Aim of C++ class is to provide programmer a means for creating new types that can be used as conveniently as built-in types
- C++ class is much more powerful than structure in C
  - not just data members
  - add member functions
  - add access control mechanism
  - Define a class => Define a **new type !**

Class Definitions

- Example:

```
class Complex{              // name of class type
private:                    // access specifier
  int realPart;             // data member declaration
  int imaginaryPart;        // data member declaration
public:                     // access specifier
  Complex(int=0,int=0);     // constructor
  ~Complex();               // destructor
  int get_realPart();       // member function declaration
  int get_imaginaryPart();  // member function declaration
};
```

Private vs. Public (1/3)

- Private:
  - Can't access outside the class scope

```
int main(){
    Complex comp;
    cout << comp.realPart << endl;  // error!!
    return 0;
}
```

  - Can access inside the class scope

```
int Complex::get_realPart() {return realPart;}
```

Private vs. Public (2/3)

- Public:
  - Can access through dot (.) or arrow (->) outside the class scope
    - E.g.,

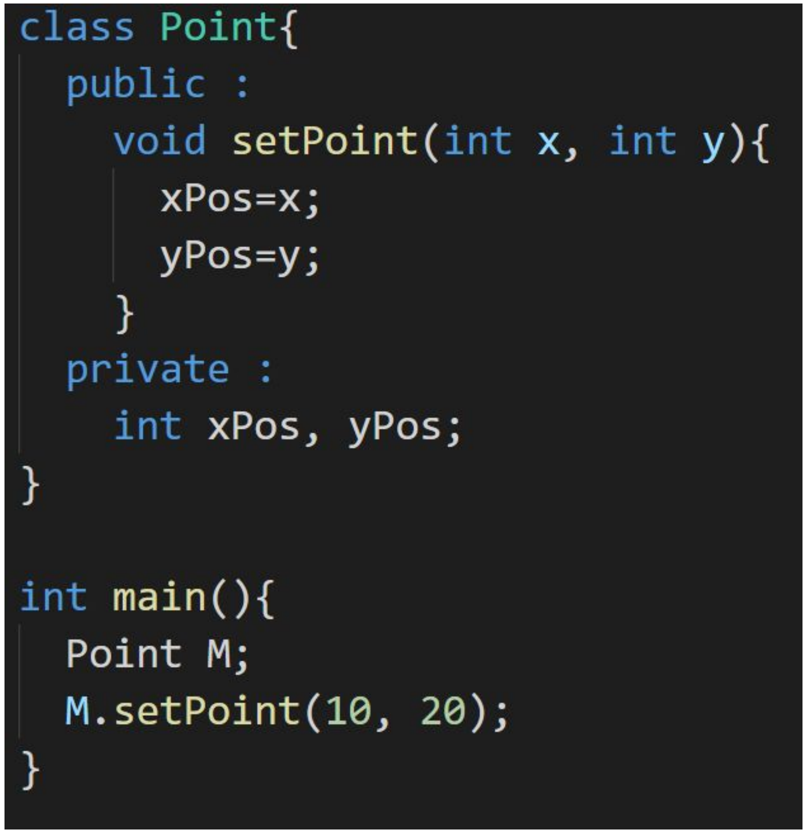
```
int main(){
    Complex comp;
    cout << comp.get_realPart() << endl;
    return 0;
}
```

Private vs. Public (3/3)

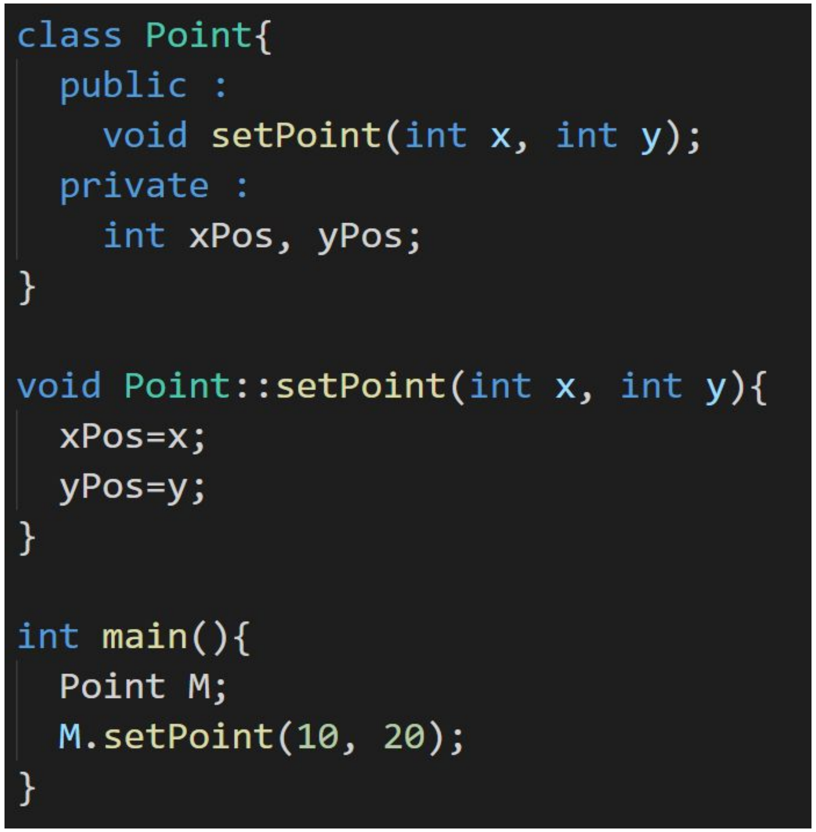
- Both data members and member functions can be either private or public
- If you didn't specify, the **default type is private**.
- Data members are usually private
  - you do not know exact representation
  - object is manipulated through member functions
- Member functions are usually public
  - you can use public interface for object manipulations
  - you still do not know how these member functions get implemented

Member Function

- We can implement the member function inside the class or outside the class.



```
class Point{
  public :
    void setPoint(int x, int y){
      xPos=x;
      yPos=y;
    }
  private :
    int xPos, yPos;
}

int main(){
  Point M;
  M.setPoint(10, 20);
}
```



```
class Point{
  public :
    void setPoint(int x, int y);
  private :
    int xPos, yPos;
}

void Point::setPoint(int x, int y){
  xPos=x;
  yPos=y;
}

int main(){
  Point M;
  M.setPoint(10, 20);
}
```

Member Function

- Example:

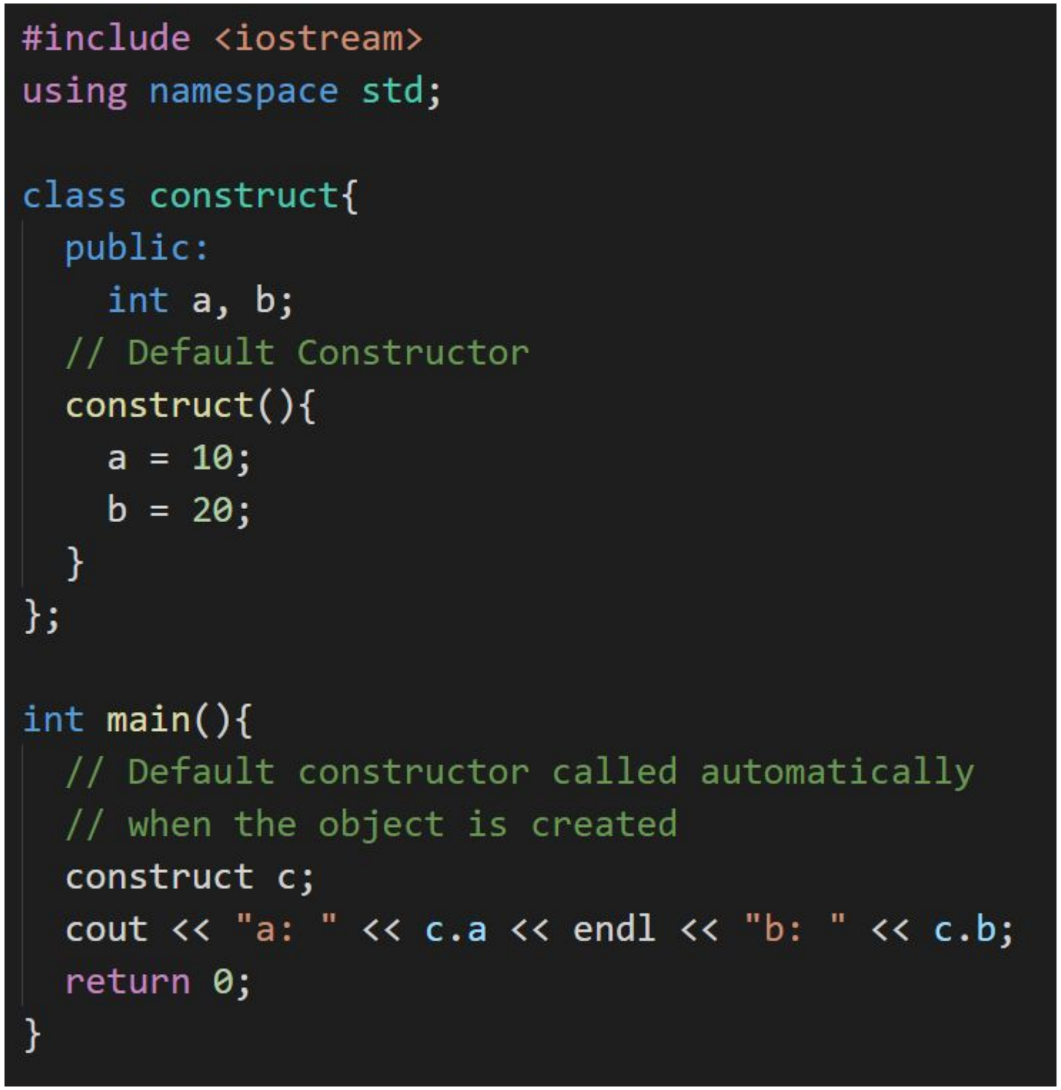
```
int Complex::get_realPart(){
    ……
}
```

- Like other function definitions
  - place outside class definition
  - must specify the class it belongs to
    - different classes can have member functions with same name
  - :: is called scope resolution operator

Constructor

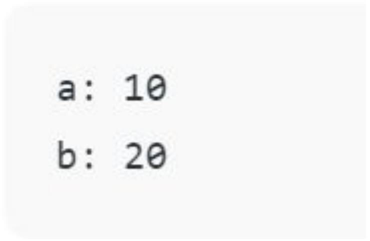
- A constructor is a special type of member function of a class which **initializes objects of a class**.
- In C++, constructor is **automatically called** when object(instance of class) is created. It is special member function of the class because it does not have any return type.
- A class can have many kinds of constructors.

Default Constructor



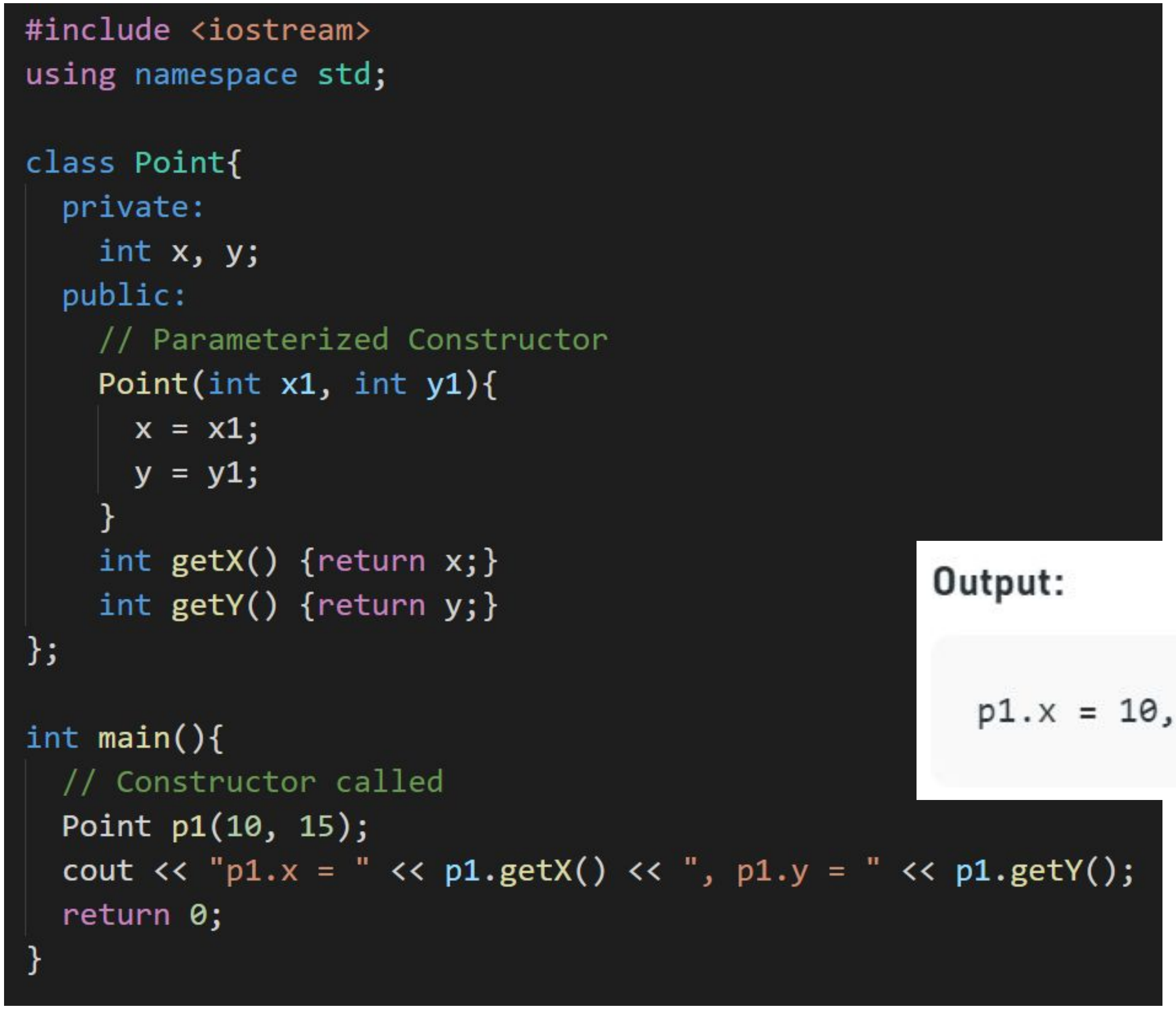
```
#include <iostream>
using namespace std;

class construct{
  public:
    int a, b;
  // Default Constructor
  construct(){
    a = 10;
    b = 20;
  }
};

int main(){
  // Default constructor called automatically
  // when the object is created
  construct c;
  cout << "a: " << c.a << endl << "b: " << c.b;
  return 0;
}
```

Output:

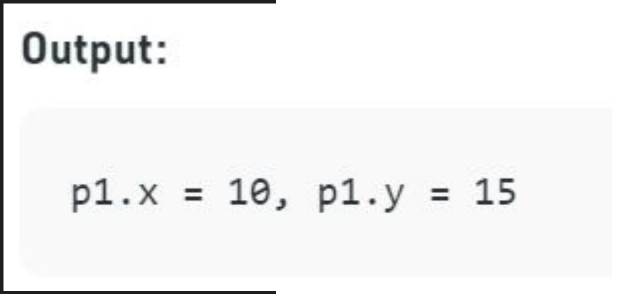


```
a: 10
b: 20
```

Parameterized Constructor



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

class Point{
  private:
    int x, y;
  public:
    // Parameterized Constructor
    Point(int x1, int y1){
      x = x1;
      y = y1;
    }
    int getX() {return x;}
    int getY() {return y;}
};

int main(){
  // Constructor called
  Point p1(10, 15);
  cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();
  return 0;
}
```

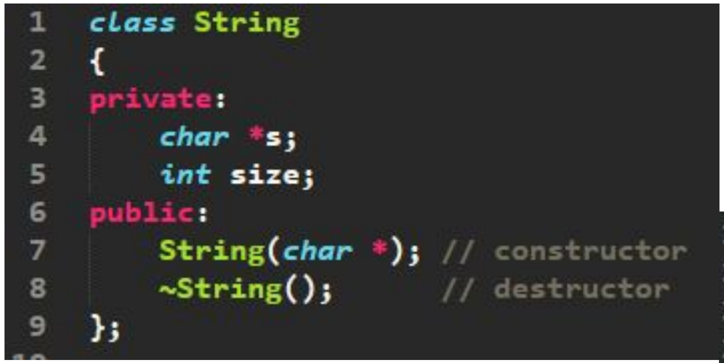


Output:

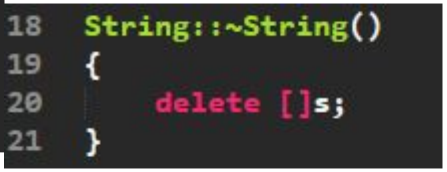
```
p1.x = 10, p1.y = 15
```

Destructor

- Example:



```
class String
{
private:
    char *s;
    int size;
public:
    String(char *); // constructor
    ~String();      // destructor
};
```



```
String::~String()
{
    delete []s;
}
```

When a class **contains a pointer to memory allocated in class**, we should write a destructor to release memory before the class instance is destroyed

- Destructor is called when objects leave scope
- Can not have the return type
- Can not have any parameter
- Performs termination housekeeping before the system reclaims the object's memory

Passing Class as Argument

- Use class as a new type
- Example:

```
Complex add_two_complexes(Complex a, Complex b){
  ……
}
```

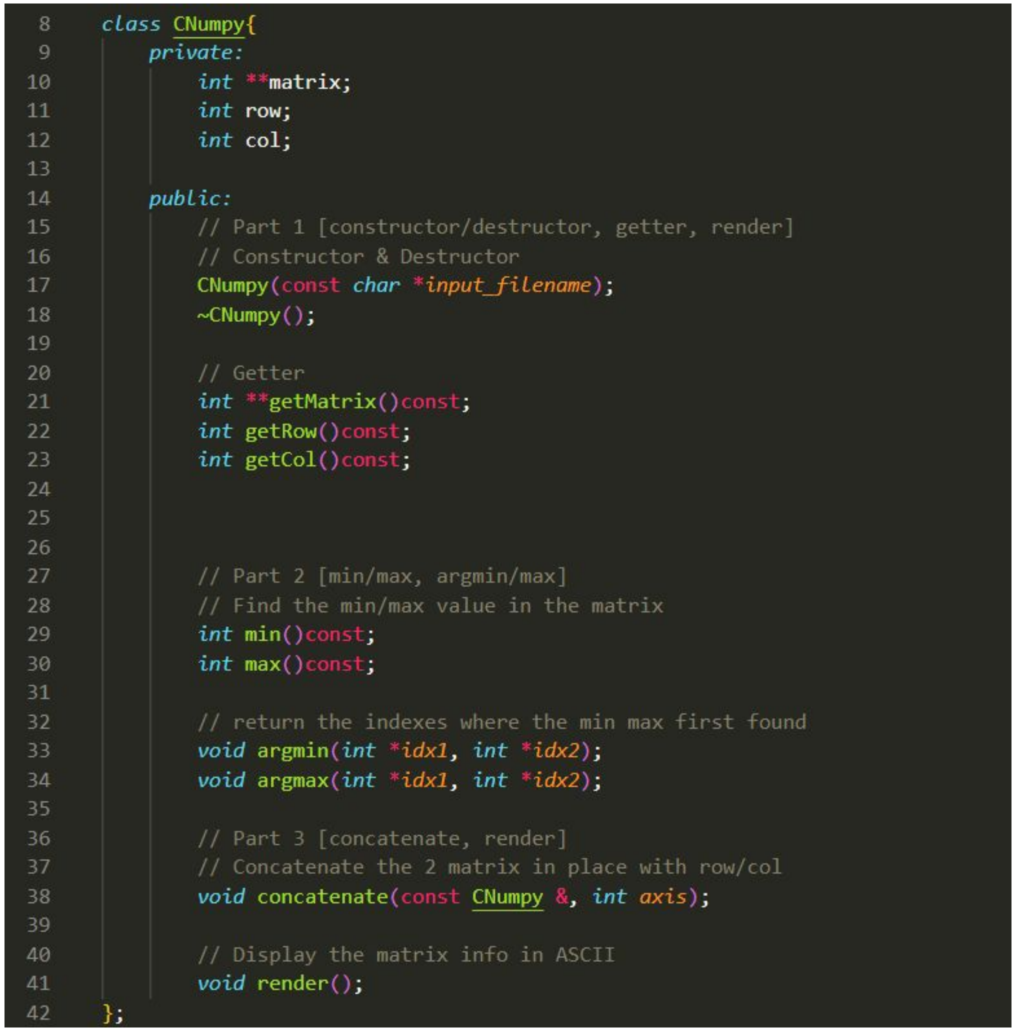
- Return type: Complex
- Parameter a’s type: Complex
- Parameter b’s type: Complex

Lab exercise

NumPy, short for Numerical Python, is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

In this Lab, you will implement simple numpy method in c++ to practice the class in OOP. In this problem, you should use 2D arrays in c++ to implement some functions of array.

CNumpy.h

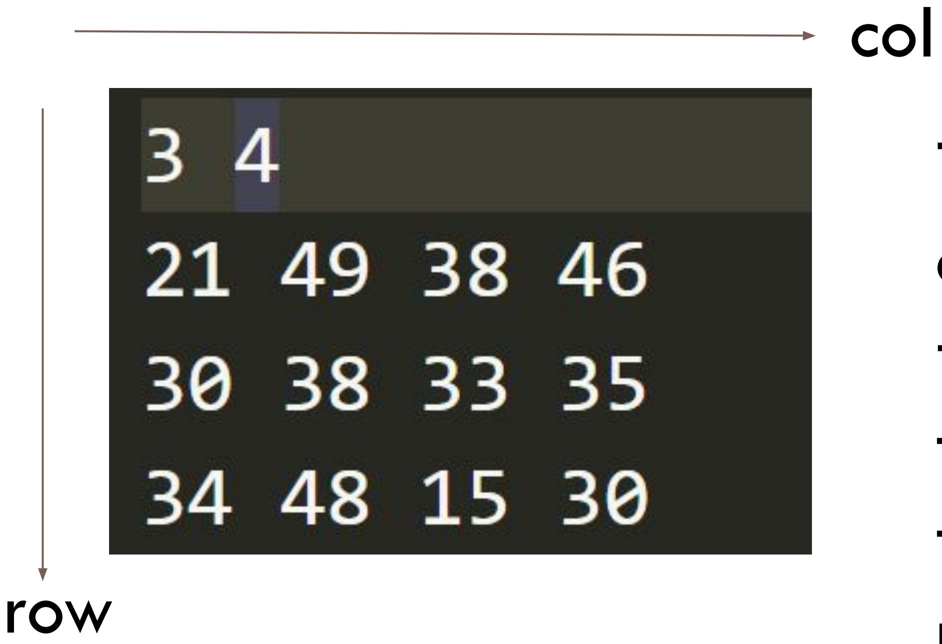


```
class CNumpy{
    private:
        int **matrix;
        int row;
        int col;

    public:
        // Part 1 [constructor/destructor, getter, render]
        // Constructor & Destructor
        CNumpy(const char *input_filename);
        ~CNumpy();

        // Getter
        int **getMatrix()const;
        int getRow()const;
        int getCol()const;



        // Part 2 [min/max, argmin/max]
        // Find the min/max value in the matrix
        int min()const;
        int max()const;

        // return the indexes where the min max first found
        void argmin(int *idx1, int *idx2);
        void argmax(int *idx1, int *idx2);

        // Part 3 [concatenate, render]
        // Concatenate the 2 matrix in place with row/col
        void concatenate(const CNumpy &, int axis);

        // Display the matrix info in ASCII
        void render();
};
```

Given CNumpy.h file, you should create CNumpy.cpp to implement the method define in this class.

There are 3 parts in this Lab.

Part1 constructor/destructor, getter

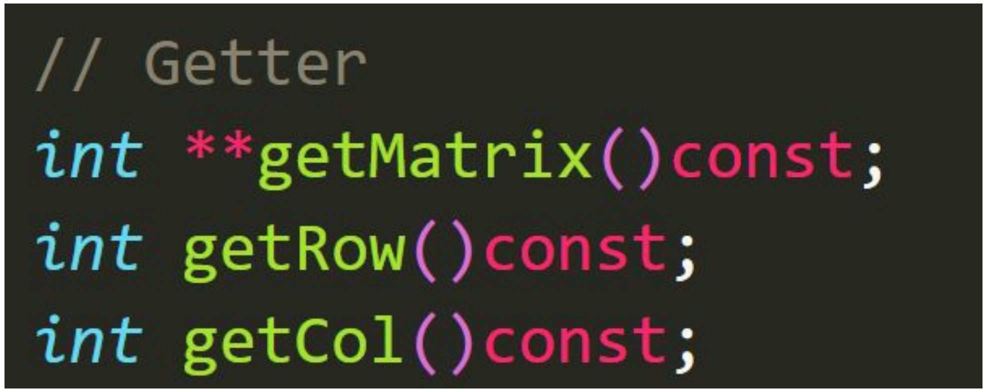


```
3 4
21 49 38 46
30 38 33 35
34 48 15 30
```

The constructor reads a file containing matrix data. The first line of the file contains two numbers representing the rows and columns of this matrix. Dynamic allocation is used to allocate memory, and to parse the values of the matrix.

Part1constructor/destructor, getter

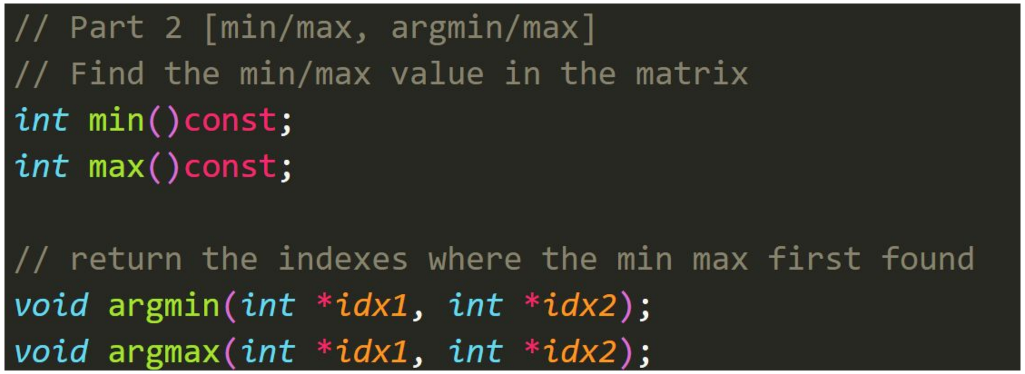
The destructor will also need to be implemented to free the memory before the program exit. TA will also use valgrind to check the program, if there exists the memory leak, you will also fail in this Lab.



```
// Getter
int **getMatrix()const;
int getRow()const;
int getCol()const;
```

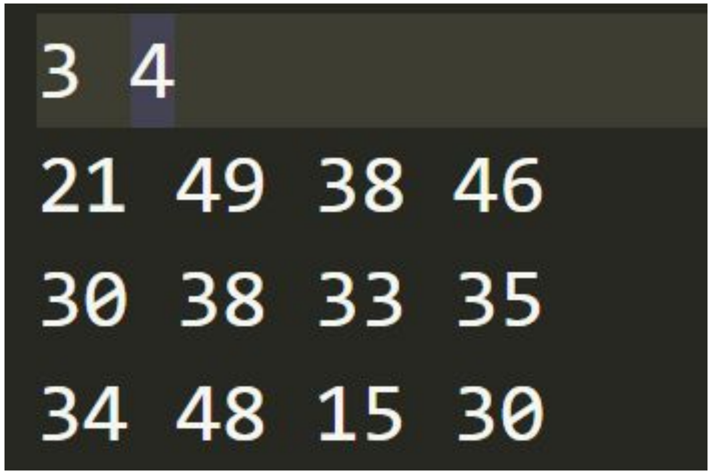
The getter simply returns the private member to the main program. The const keyword is specified to allow this method to be used by const objects.

Part 2 Min/Max Operation



```
// Part 2 [min/max, argmin/max]
// Find the min/max value in the matrix
int min()const;
int max()const;

// return the indexes where the min max first found
void argmin(int *idx1, int *idx2);
void argmax(int *idx1, int *idx2);
```

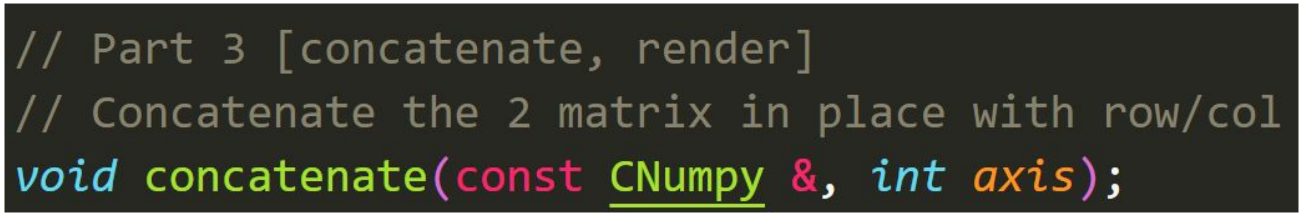
Find the min/max value in the 2d array, and also the index that first found in the 2d array.



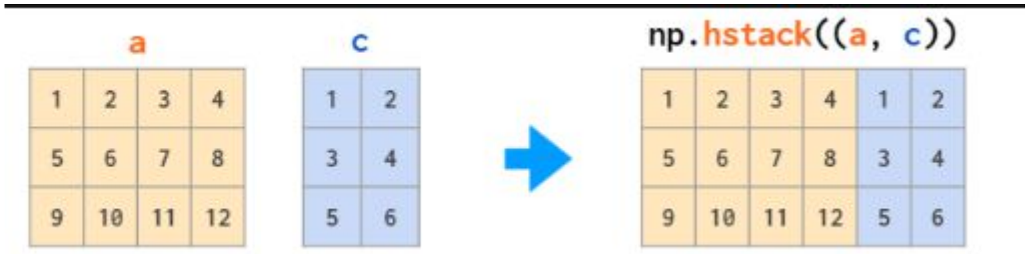
```
3 4
21 49 38 46
30 38 33 35
34 48 15 30
```

ex: min = 15,
happens at (2, 2) [argmin]
change idx1, idx2 to (2, 2)

Part 3 Matrix Concatenate/Render

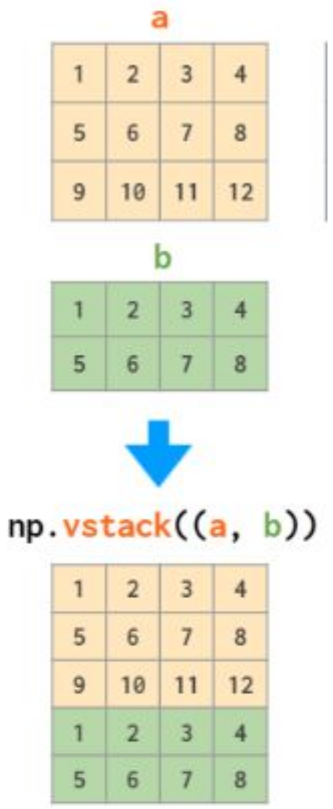


```
// Part 3 [concatenate, render]
// Concatenate the 2 matrix in place with row/col
void concatenate(const CNumpy &, int axis);
```



a				c	
1	2	3	4	1	2
5	6	7	8	3	4
9	10	11	12	5	6

axis=1

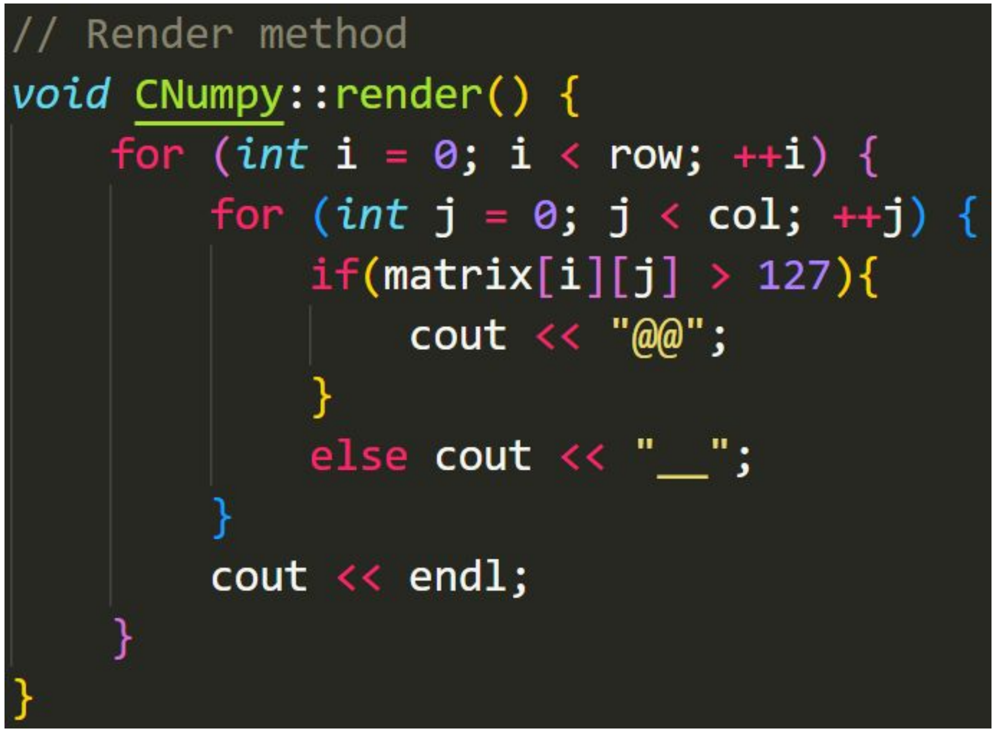


1	2	3	4
5	6	7	8
9	10	11	12
1	2	3	4
5	6	7	8

Concatenate 2 matrix data in place base on the axis.

axis=0

Part 3 Matrix Concatenate/Render



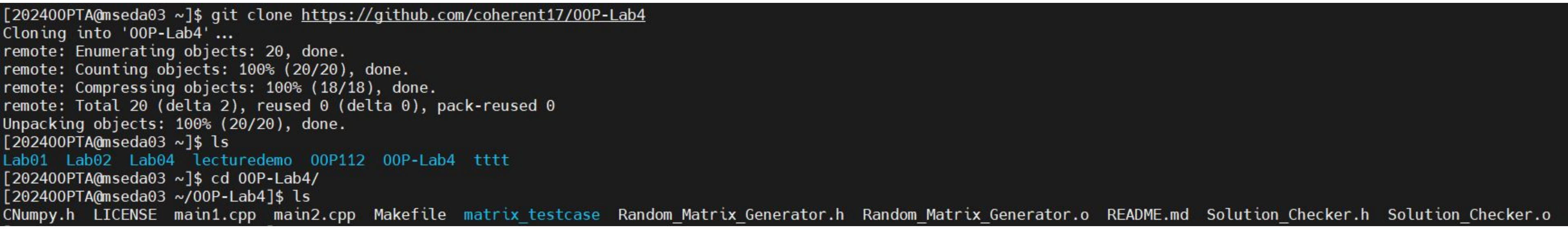
```cpp
// Render method
void CNumpy::render() {
    for (int i = 0; i < row; ++i) {
        for (int j = 0; j < col; ++j) {
            if(matrix[i][j] > 127){
                cout << "@@";
            }
            else cout << "__";
        }
        cout << endl;
    }
}
```

To visualize the matrix data, we can use ASCII ART to represent the matrix data. You can just copy this code into CNumpy.cpp

Get the code template

git clone https://github.com/coherent17/OOP-Lab4

cd OOP-Lab4



```
[202400PTA@mseda03 ~]$ git clone https://github.com/coherent17/OOP-Lab4
Cloning into 'OOP-Lab4'...
remote: Enumerating objects: 20, done.
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 20 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (20/20), done.
[202400PTA@mseda03 ~]$ ls
Lab01  Lab02  Lab04  lecturedemo  OOP112  OOP-Lab4  tttt
[202400PTA@mseda03 ~]$ cd OOP-Lab4/
[202400PTA@mseda03 ~/OOP-Lab4]$ ls
CNumpy.h  LICENSE  main1.cpp  main2.cpp  Makefile  matrix_testcase  Random_Matrix_Generator.h  Random_Matrix_Generator.o  README.md  Solution_Checker.h  Solution_Checker.o
```

Create CNumpy.cpp and finish the member function as mentioned before.

You can not modify main1.cpp main2.cpp and CNumpy.h

Compile & Run

[Compile]

```
g++ -g -Wall -c CNumpy.cpp
g++ -g -Wall main1.cpp CNumpy.o Random_Matrix_Generator.o Solution_Checker.o -o Lab4-1
g++ -g -Wall main2.cpp CNumpy.o Random_Matrix_Generator.o Solution_Checker.o -o Lab4-2
```

or simply type make

[Execute]

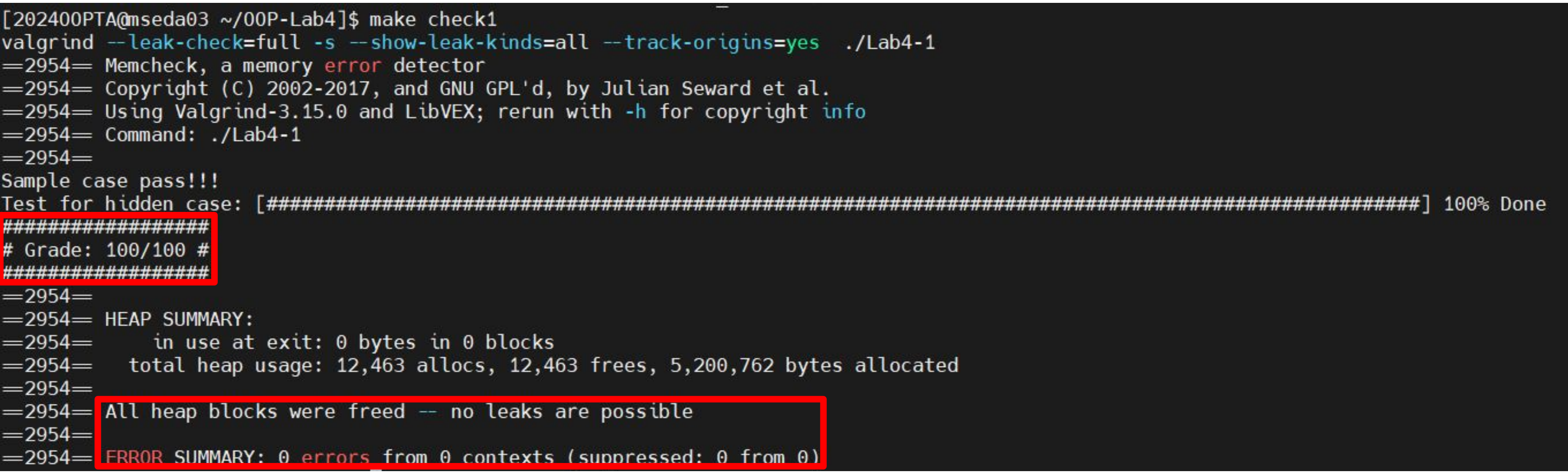
```
./Lab4-1 [for part1 & part2]
./Lab4-2 [for part3]
```

[Valgrind]

```
valgrind --leak-check=full -s --show-leak-kinds=all --track-origins=yes ./Lab4-1
valgrind --leak-check=full -s --show-leak-kinds=all --track-origins=yes ./Lab4-2
```

or simply type make check1/ make check2

Demo Part1 & 2

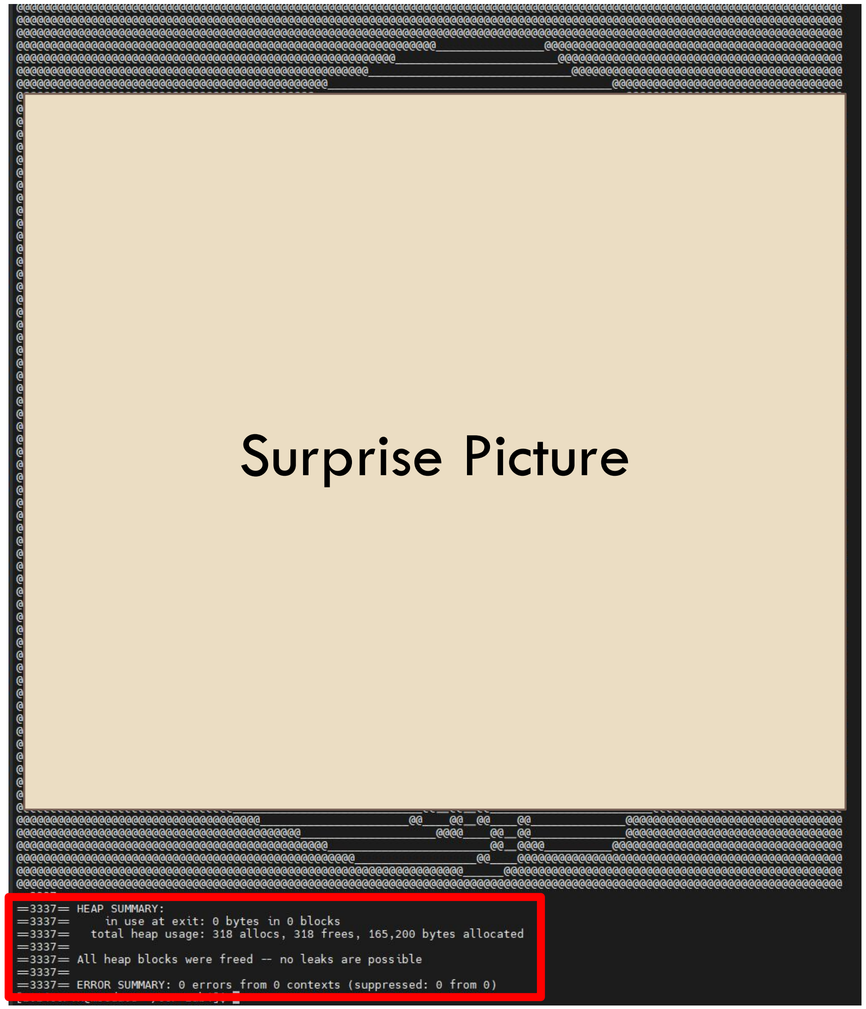


```
[202400PTA@mseda03 ~/OOP-Lab4]$ make check1
valgrind --leak-check=full -s --show-leak-kinds=all --track-origins=yes  ./Lab4-1
==2954== Memcheck, a memory error detector
==2954== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2954== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==2954== Command: ./Lab4-1
==2954==
Sample case pass!!!
Test for hidden case: [####################################################################################################] 100% Done
##################
# Grade: 100/100 #
##################
==2954==
==2954== HEAP SUMMARY:
==2954==     in use at exit: 0 bytes in 0 blocks
==2954==   total heap usage: 12,463 allocs, 12,463 frees, 5,200,762 bytes allocated
==2954==
==2954== All heap blocks were freed -- no leaks are possible
==2954==
==2954== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Get 100/100.

No memory leak is allowed.

Demo Part3



```
==3337== HEAP SUMMARY:
==3337==     in use at exit: 0 bytes in 0 blocks
==3337==   total heap usage: 318 allocs, 318 frees, 165,200 bytes allocated
==3337==
==3337== All heap blocks were freed -- no leaks are possible
==3337==
==3337== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Ask the TA for demo.

And answer who is the person in the surprise picture.

Also, no memory leak.

If there exists the memory leak, you fail the lab also!!

Submission

- Ask TAs for demo (both part1/2/3)
- Try your best to debug your code by yourself
- Upload all your cpp to new E3 and zip them together
- Naming rule : studentID_lab4.zip