CPSC 131
Data Structures

Dr. Shilpa Lakhanpal
shlakhanpal@fullerton.edu

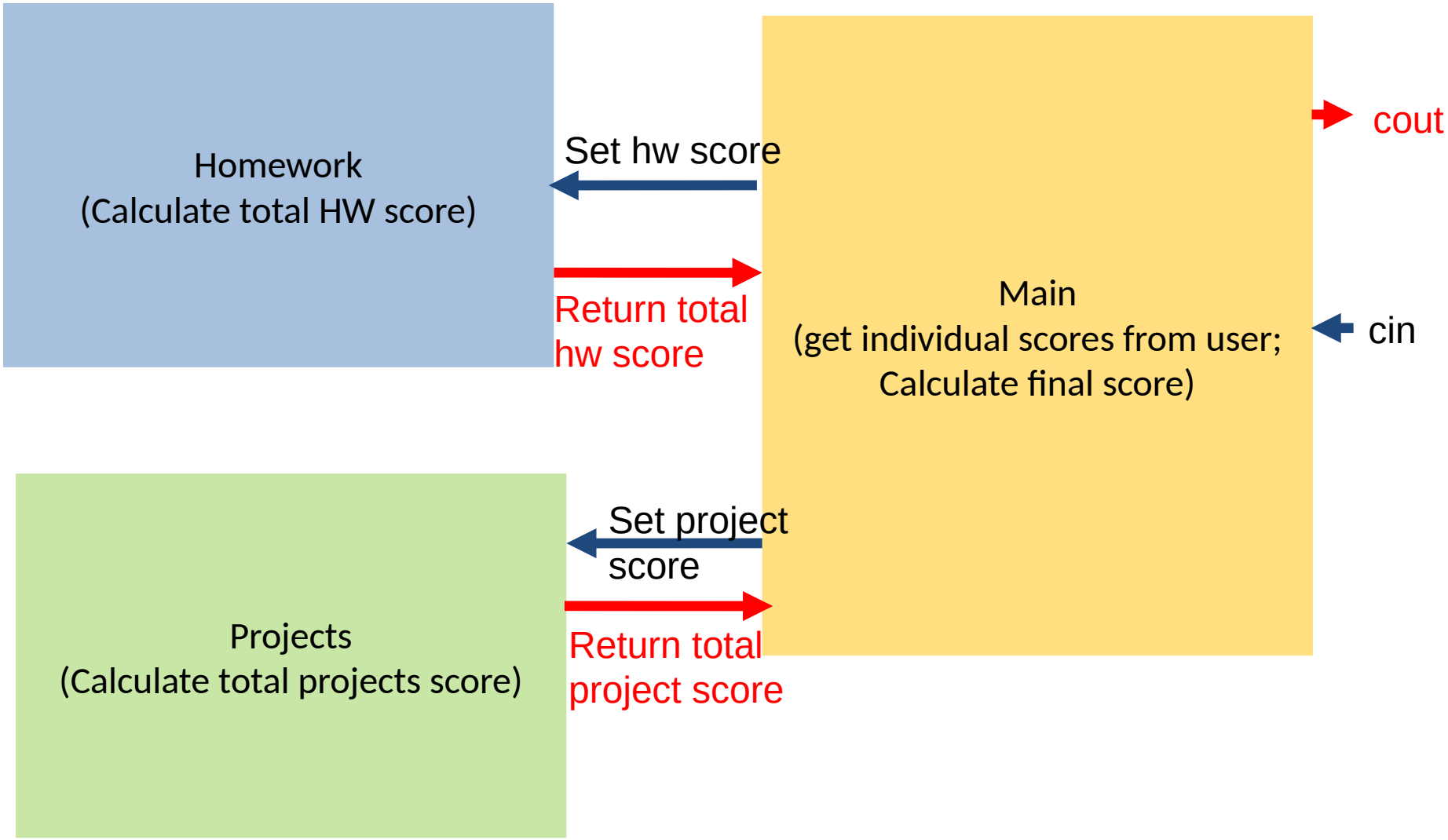
Let’s write some code

Write a C++ program to compute course total from eight homework scores and four project scores

Without classes

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

int main() {
     int hw[8];
     cout << "Enter homework scores: ";
     for (int i = 0; i < 8; i++) {
     cin >> hw[i];
     }
     int hwTotal = 0;
     for (int i = 0; i < 8; i++)
     hwTotal += hw[i];
     cout << "Total of hw assignments: " << hwTotal << endl;
     int project[4];
     cout << "Enter project scores: ";
     for (int i = 0; i < 4; i++) {
     cin >> project[i];
     }
     int projectTotal = 0;
     for (int i = 0; i < 4; i++)
     projectTotal += project[i];
     cout << "Total of projects: " << projectTotal << endl;
     int courseTotal = hwTotal + projectTotal;
     cout << "Course total: " << courseTotal << endl;
     return 0;
     }
```

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

int main() {
        int hw[8];
        cout << "Enter homework scores: ";
        for (int i = 0; i < 8; i++) {
                cin >> hw[i];
        }
        int hwTotal = 0;
        for (int i = 0; i < 8; i++)
                hwTotal += hw[i];
        cout << "Total of hw assignments: " << hwTotal << endl;
        int project[4];
        cout << "Enter project scores: ";
        for (int i = 0; i < 4; i++) ++) {
                cin >> project[i];
        }
        int projectTotal = 0;
        for (int i = 0; i < 4; i++)
                projectTotal += project[i];
        cout << "Total of projects: " << projectTotal << endl;
        int courseTotal = courseTotal = hwTotal + projectTotal;
        cout << "Course total: " << courseTotal << endl;
        return 0;
        }
```


Homework
(Calculate total HW score)
Set hw score
Return total hw score
Main
(get individual scores from user; Calculate final score)
cout
cin
Projects
(Calculate total projects score)
Set project score
Return total project score

Not object-oriented

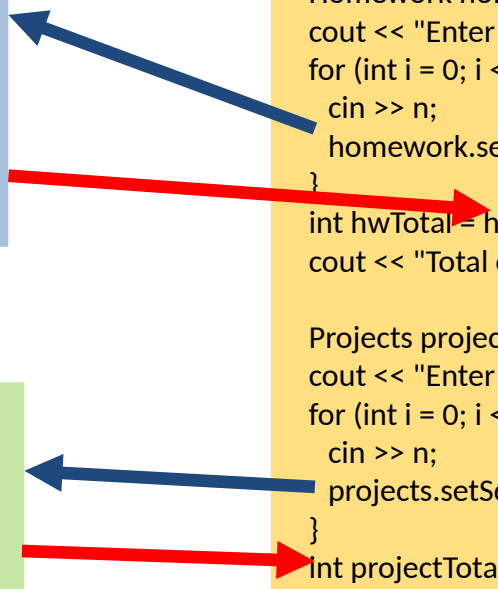
```
#include <iostream>



using namespace std;

int main() {

        int hw[8];
        cout << "Enter homework scores: ";
        for (int i = 0; i < 8; i++) {
                cin >> hw[i];
        }


        int hwTotal = 0;
        for (int i = 0; i < 8; i++)
                hwTotal += hw[i];

        cout << "Total of hw assignments: " <<
        hwTotal << endl;


  // ...
```

Object-oriented

```
#include <iostream>
#include "Homework.h"



using namespace std;

int main() {
  int n;
  Homework homework;
  cout << "Enter homework scores: ";
  for (int i = 0; i < 8; i++) {
    cin >> n;
    homework.setScore(i,n);
  }

  int hwTotal = homework.getTotal();



  cout << "Total of hw assignments: " << hwTotal <<
  endl;

  // ...
```

Homework
(Calculate total HW score)

Projects
(Calculate total projects score)



```
#include <iostream>
#include "Homework.h“
#include “Projects.h“

using namespace std;
int main() {
  int n;
  Homework homework;
  cout << "Enter homework scores: ";
  for (int i = 0; i < 8; i++) {
    cin >> n;
    homework.setScore(i,n);
  }
  int hwTotal = homework.getTotal();
  cout << "Total of hw assignments: " << hwTotal << endl;

  Projects projects;
  cout << "Enter project scores: ";
  for (int i = 0; i < 4; i++) {
    cin >> n;
    projects.setScore(i,n);
  }
  int projectTotal = projects.getTotal();
  cout << "Total of projects: " << projectTotal << endl;

  double courseTotal = hwTotal + projectTotal;
  cout << "Course total: " << courseTotal << endl;
  system("pause");
  }
```

Main.cpp

```
#include <iostream>
#include "Homework.h“
#include “Projects.h“

using namespace std;
int main() {
  int n;
  Homework homework;
  cout << "Enter homework scores
  for (int i = 0; i < 8; i++) {
    cin >> n;
    homework.setScore(i,n);
  }
  int hwTotal = homework.getTotal
  cout << "Total of hw assignments
  hwTotal << endl;

  Projects projects;
  cout << "Enter project scores: ";
  for (int i = 0; i < 4; i++) {
    cin >> n;
    projects.setScore(i,n);
  }
  int projectTotal = projects.getTot
  cout << "Total of projects: " <<
  projectTotal << endl;

  double courseTotal = hwTotal +
  projectTotal;
  cout << "Course total: " << courseTotal
  << endl;
  system("pause");
  }
```

Homework.h

```
#pragma once
class Homework
{
public:
  Homework();
  ~Homework();
  void setScore(int i, int n);
  int getTotal();
  private:
  int project[10];
  };
```

Homework.cpp

```
#include "Homework.h"

Homework::Homework() {
  for (int i = 0; i < 10; i++)
     hw[i] = 0;
  }

etScore(int i, int n) {

Total() {

 i++)

ework() {}
```

Source (.cpp) files “include” header files (.h)

Typically:

- Header files contain **declarations**
  - Introduces a variable or function
  - A “contract”
- Source files contain **definitions**
  - Implements *previously declared* functions
  - Fulfills the contract

```
Projects::Projects() {
  for (int i = 0; i < 10; i++)
     project[i] = 0;
  }

  // ...
```

Projects.cpp

Homework.h

```
#pragma once
class Homework
{
public:
  Homework();
  ~Homework();
  void setScore(int i,int n);
  int getTotal();

  private:
  int hw[10];
  };
```

Member functions/methods

Member variables/ data members

Constructor

Private
(only accessible from within the class definition)

Homework.cpp

```
#include "Homework.h"

Homework::Homework() {
  for (int i = 0; i < 10; i++)
     hw[i] = 0;
  }

  void Homework::setScore(int i,int n) {
  hw[i] = n;
  }

  int Homework::getTotal() {
  int total = 0;
  for (int i = 0; i < 10; i++)
     total += hw[i];
  return total;
  }

  Homework::~Homework() {}
```

Three types of constructors

- Default constructor
  - No inputs to constructor
- Constructor with inputs for initialization
- Copy constructor

Default constructor

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

class construct {
public:
	int a, b;
	// Default Constructor
	construct()
	{
		a = 15;
		b = 20;
	}
};
int main()
{
	// Default constructor called automatically when the object is created
	construct c;
	cout << "a: " << c.a << endl
		<< "b: " << c.b;
	return 1;
}
```

Constructor with inputs for initialization

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

class Point {
private:
	int x, y;

public:
	// Constructor
	Point(int x1, int y1)
	{
		x = x1;
		y = y1;
	}

	int getX()
	{
		return x;
	}
	int getY()
	{
		return y;
	}
};
```

```
int main()
{
	Point p1(10, 15);

	cout << "p1.x = " <<
p1.getX() << ", p1.y = " <<
p1.getY();

	return 0;
}
```

Copy Constructor

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

class Point
{
private:
        int x, y;
public:
        Point(int x1, int y1) { x = x1; y = y1; }

        // Copy constructor
        Point(const Point& p) {x = p.x; y =
p.y; }

        int getX()    {  return x; }
        int getY()    {  return y; }
};
```

```
int main()
{
Point p1(10, 15); // Normal
constructor is called here
Point p2 = p1; // Copy
constructor is called here

// Let us access values
assigned by constructors
cout << "p1.x = " << p1.getX()
<< ", p1.y = " << p1.getY();

cout << "\n p2.x = " <<
p2.getX() << ", p2.y = " <<
p2.getY();

return 0;
}
```

Assignment Operator

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

class Point
{
private:
        int x, y;
public:
        Point(int x1, int y1) { x = x1; y = y1; }

        // Assignment Operator
        Point& operator = (const Point& p) {
                if (this != &p) { x = p.x; y = p.y;}
        return *this;}

        int getX()    {  return x; }
        int getY()    {  return y; }
};
```

```
int main()
{
Point p1(10, 15), p2(56,87); //
Normal constructor is called
here

// Let us access values
assigned by constructors
cout << "p1.x = " << p1.getX()
<< ", p1.y = " << p1.getY();

cout << "\n p2.x = " <<
p2.getX() << ", p2.y = " <<
p2.getY();

p2 = p1; // Assignment
operator is called here

cout << "\n p2.x = " <<
p2.getX() << ", p2.y = " <<
p2.getY();
return 0;
}
```

Copy Constructor vs Assignment operator

- Copy constructor is called when a new object is created from an existing object, as a copy of the existing object
- Assignment operator is called when an already initialized object is assigned a new value from another existing object

Homework.h

```
#pragma once
class Homework
{
public:
  Homework();
  Homework(const string &nm);
  ~Homework();
  void setScore(int i,int n);
  void setName(const string &nm)
  int getTotal();

  private:
  int hw[10];
  string name;
  };
```

Homework.cpp

```
#include "Homework.h"

Homework::Homework() {
  name = “fnu”;
  for (int i = 0; i < 10; i++)
     hw[i] = 0;
  }

  Homework::Homework (const string &nm) {
    name = nm;
  for (int i = 0; i < 10; i++)
     hw[i] = 0;
  }

  void Homework::setName (const string &nm) {
    name = nm;
  }

  void Homework::setScore(int i,int n) {
  hw[i] = n;
  }

  int Homework::getTotal() {
  int total = 0;
  for (int i = 0; i < 10; i++)
     total += hw[i];
  }

  Homework::~Homework() {}
```

Main.cpp

```
#include <iostream>
#include "Homework.h“

using namespace std;
int main() {
   int n;
   Homework john_hw;
   john_hw.setName(“John”);

   // …

   Homework jane_hw(“Jane”);
   // …
   Homework section01_hw[28];
   section01_hw[0].setName(“Pamela”);

   }
```

Changing the code to solve slightly different problems

- Which of these files
  1. Main.cpp
  2. Homework.h
  3. Homework.cpp
  4. Projects.h
  5. Projects.cpp
- Should be changed to
  - Increase number of homework assignments
  - Change totalScore to discard lowest score
  - Use a GUI instead of keyboard/terminal input
  - Use dynamic arrays

References

- CSUF CPSC 131 Slides: Object Oriented Design, Dr. Anand Panangadan