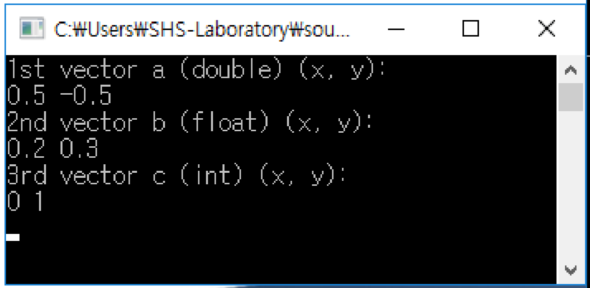
LAB I

Week 12

Seoul National University
Graphics & Media Lab
HyeonSeung Shin

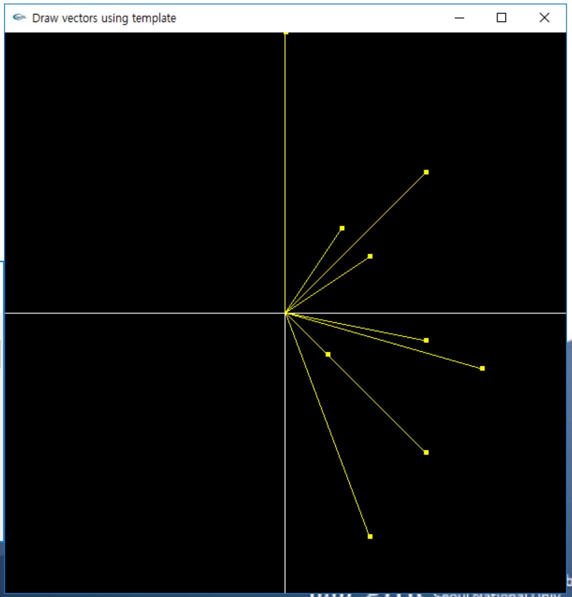
Today's Mission

- We extend the last week's operator overloading using template.
- Input three 2D vectors **a** (double), **b** (float), and **c** (int) in the range [-1, 1] x [-1, 1].
- Draw the three input vectors in the OpenGL window.
- As the user press the last enter key, draw
  - **a** + **b** (The return type is determined left operand)
  - **a** – **b**
  - **a** * (**b** · **c**)
  - **a** += **c**
  - **a** -= **b**
  - Vec2(**a**[0] + **b**[0], **a**[1] - **b**[1])



C:₩Users₩SHS-Laboratory₩sou...

```
1st vector a (double) (x, y):
0.5 -0.5
2nd vector b (float) (x, y):
0.2 0.3
3rd vector c (int) (x, y):
0 1
```

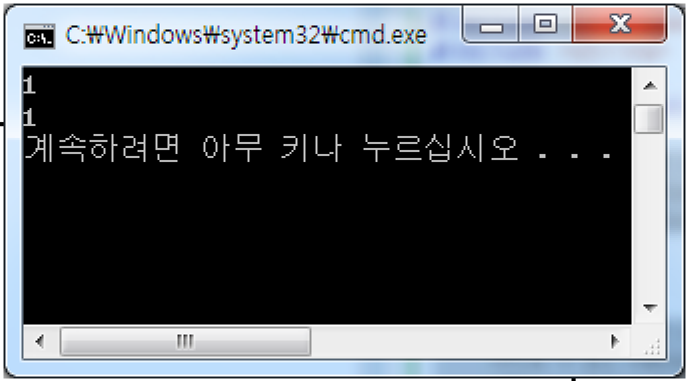


Motivation: Avoid Code Duplication

- Example

```
int compare(const string& a, const string& b) {
   if(a < b) return -1;
   if(a > b) return +1;
   return 0;
}

int compare(const double& a, const double& b) {
   if(a < b) return -1;
   if(a > b) return +1;
   return 0;
}

void main() {
   const string a("Program"), b("Methodology");
   const double c(2.0), d(1.0);

   cout << compare(a, b) << endl;
   cout << compare(c, d) << endl;
}
```

Duplication !



```
1
1
계속하려면 아무 키나 누르십시오 . . .
```

Function Template

Template Parameter List

```
template<typename T>
int compare(const T& a, const T& b) {
   if(a < b) return -1;
   if(a > b) return +1;
   return 0;
}
```

- Actual instantiation of T is determined based on how the function is called.
  - T can be `int` or `double` or `std::string` or …

Template Argument for Non-type Parameters

- Definition of class Vec for arbitrary dimension

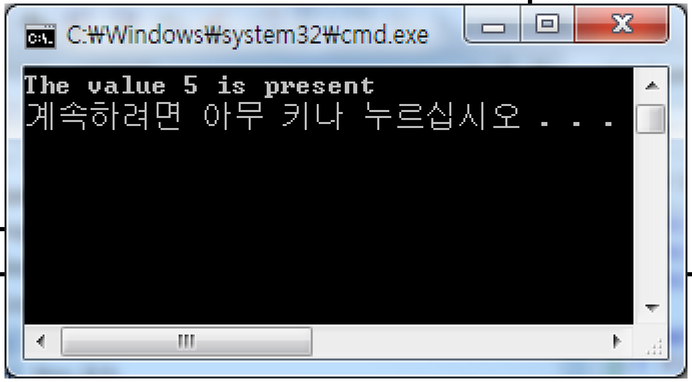
```
template<typename T, int DIM>
class Vec {
public :
   Vec() {}

   T val[DIM];
};

void main() {
   Vec<float, 2>      a;     // 2-dimensional float vector
   Vec<double, 3>     b;     // 3-dimensional double vector
}
```

The Complete Vec2 Template Class

```
template<typename T>
class Vec2 {
public :
   Vec2() {}
   Vec2(const T& a, const T& b) { val[0] = a; val[1] = b; }

   template<typename S> void set(const Vec2<S>&);
   void set(const T& a, const T& b);

   const T& operator[](std::size_t i) const;

   T val[2];
};

template<typename T> template<typename S>
void Vec2<T>::set(const Vec2<S>& v) { val[0] = v.val[0]; val[1] = v.val[1]; }

template<typename T>
void Vec2<T>::set(const T& a, const T& b) { val[0] = a; val[1] = b; }

template<typename T>
T& Vec2<T>::operator[](std::size_t i) const { return val[i]; }

template<typename T>
Vec2<T> operator+(const Vec2<T>& a, const Vec2<T>& b) {
   return Vec2<T>(a.val[0] + b.val[0], a.val[1] + b.val[1]);
}
```

An Example of Simple Generic Algorithm

```
template<typename T, typename S>
T find(T begin, T end, S val) {
   for(T it = begin ; it != end ; it++) {
      if(*it == val)
         return it;
   }
   return end;
}
```

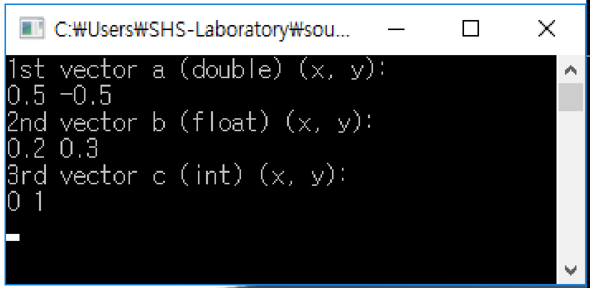


```
C:\Windows\system32\cmd.exe
The value 5 is present
계속하려면 아무 키나 누르십시오 . . .
```

```
void main() {
   int ia[5] = { 1,2,3,4,5 };
   int val = 5;

   int * result = find(ia,ia+5,val); // T is int*, S is int

   cout << "The value " << val
        << (result == ia+5 ? " is not present"
          : " is present") << endl;
}
```

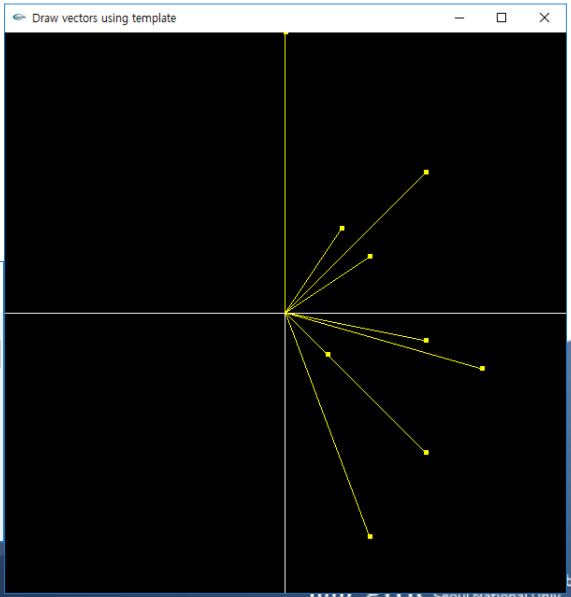
Today's Mission

- We extend the last week's operator overloading using template.
- Input three 2D vectors **a** (double), **b** (float), and **c** (int) in the range [-1, 1] x [-1, 1].
- Draw the three input vectors in the OpenGL window.
- As the user press the last enter key, draw
  - **a** + **b** (The return type is determined left operand)
  - **a** – **b**
  - **a** * (**b** · **c**)
  - **a** += **c**
  - **a** -= **b**
  - Vec2(**a**[0] + **b**[0], **a**[1] - **b**[1])



C:₩Users₩SHS-Laboratory₩sou...

```
1st vector a (double) (x, y):
0.5 -0.5
2nd vector b (float) (x, y):
0.2 0.3
3rd vector c (int) (x, y):
0 1
```



Implementation Details

```
template<typename T>
class Vec2 {
public:
        Vec2();
        Vec2(T x, T y);
        Vec2(const Vec2& v);

        void setPos(T x, T y);
        void draw() const;

        // implement member operator overloading

private:
        T pos[2];
};

// implement non-member operator overloading

// implement  "T dotProduct(Vec2<T> a, Vec2<S> b)"
```