C++ Class Template

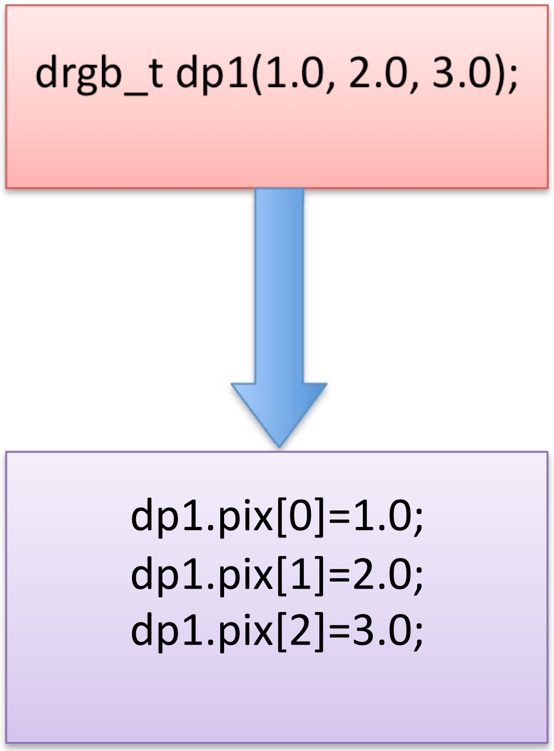
CpSc102 – Fall 2010

drgb_t

```
private:
   double pix[3];
```

drgb_t constructor overloaded

```
drgb_t(double ix=0.0, double iy=0.0, double iz=0.0)
 {
      pix[0] = ix;
      pix[1] = iy;
      pix[2] = iz;
  }
```

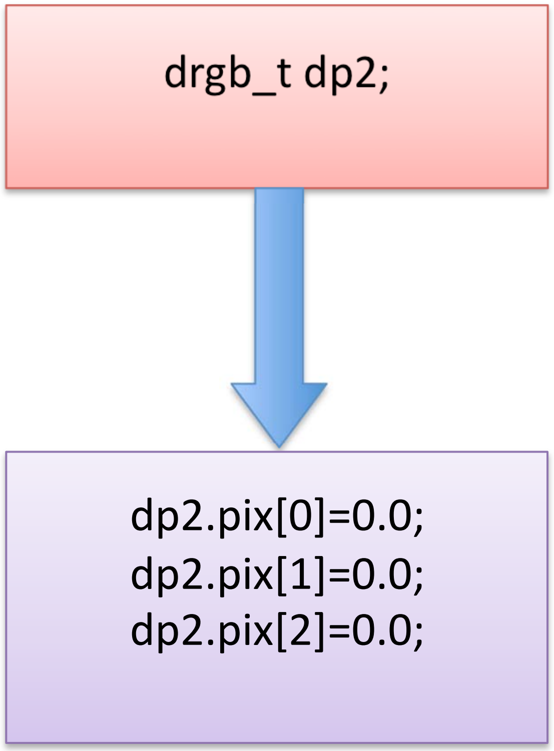


```
drgb_t dp1(1.0, 2.0, 3.0);
```

```
dp1.pix[0]=1.0;
dp1.pix[1]=2.0;
dp1.pix[2]=3.0;
```

drgb_t constructor overloaded

```
drgb_t(double ix=0.0, double iy=0.0, double iz=0.0)
 {
      pix[0] = ix;
      pix[1] = iy;
      pix[2] = iz;
  }
```



```
drgb_t dp2;
```

```
dp2.pix[0]=0.0;
dp2.pix[1]=0.0;
dp2.pix[2]=0.0;
```

drgb_t copy constructor

```
drgb_t(const drgb_t& rhs)
 {
        for(int i=0; i<3; i++) pix[i] = rhs[i];
}
```

```
drgb_t dp1(1.0, 2.0, 3.0);
drgb_t dp4(dp1);
```

$$\downarrow$$

```
dp4.pix[0]=1.0;
dp4.pix[1]=2.0;
dp4.pix[2]=3.0;
```

drgb_t assignment operator

```
drgb_t& operator=(const drgb_t& rhs)
 {
   if(this != &rhs) for(int i=0; i<3; i++) pix[i] = rhs[i];

   return *this;
 }
```

```
drgb_t  dp2(4.0, 5.0, 6.0);
drgb_t  dp3=dp4;
```

```
dp3.pix[0]=4.0;
dp3.pix[1]=5.0;
dp3.pix[2]=6.0;
```

drgb_t accessor and mutator

```
const double& operator[](int i) const
{
	return pix[i];
}
double& operator[](int i)
{
	return pix[i];
}
```

```
drgb_t  dp2(4.0, 5.0, 6.0);
dp2[1] = 3.0;
dp2[0] = dp2[2];
```

↓

```
dp2.pix[0]=6.0;
dp2.pix[1]=3.0;
dp2.pix[2]=6.0;
```

```
drgb_t dp4(1.0, 2.0, 3.0);
std::cout << "dp4 * 5 = " << dp4 * 5 << std::endl;
```

```
const drgb_t operator*(const double& s) const
{
   return drgb_t(*this) *= s;
}
```

```
drgb_t& operator*=(const double& s)
{
   for(int i=0; i<3; i++) pix[i] *= s;
   return *this;
}
```

```
friend std::ostream& operator<<(std::ostream& s, const drgb_t& rhs)
{
      return(s << rhs[0] << " " << rhs[1] << " " << rhs[2]);
}
```

```
5.0 10.0 15.0
```

drgb_t

```
friend drgb_t operator*(const double& s, const drgb_t& rhs)
 {
   drgb_t  result;
   for(int i=0; i<3; i++) result[i] = s * rhs[i];
   return result;
 }
```

```
drgb_t dp4(1.0, 2.0, 3.0);
std::cout << "3 * dp4 = " << 3 * dp4 << std::endl;
```

```
3.0 6.0 9.0
```

drgb_t

```
std::cout << "dp1 = " << dp1 << std::endl;
```

```
friend std::ostream& operator<<(std::ostream& s, const drgb_t& rhs)
{
    return(s << rhs[0] << " " << rhs[1] << " " << rhs[2]);
}
friend std::ostream& operator<<(std::ostream& s, drgb_t *rhs)
{
   return(s << (*rhs));
}
```

```
std::cout << "dp1 = " << &dp1 << std::endl;
```

drgb_t

```
drgb_t dp3;
std::cin >> dp3;
```

```
friend std::istream& operator>>(std::istream& s, drgb_t& rhs)
{
    return(s >> rhs[0] >> rhs[1] >> rhs[2]);
}
friend std::istream& operator>>(std::istream& s, drgb_t *rhs)
{
    return(s >> (*rhs));
}
```

```
drgb_t dp4;
std::cin >> &dp4;
```

drgb_t

```
...
drgb_t& operator+=(const drgb_t& rhs);
drgb_t& operator*=(const drgb_t& rhs);
const drgb_t operator+(const drgb_t& rhs) const;
const drgb_t operator*(const drgb_t& rhs) const;
bool iszero(void) const;
void clamp(double min, double max);
```

irgb_t

```
private:
    unsigned char pix[3];
```

irgb_t

```
irgb_t(unsigned char ix=0, unsigned char iy=0, unsigned char iz=0);
irgb_t(const irgb_t& rhs);
irgb_t& operator=(const irgb_t& rhs);
irgb_t& operator+=(const irgb_t& rhs);
irgb_t& operator*=(const irgb_t& rhs);
irgb_t& operator*=(const unsigned char& s);
const unsigned char& operator[](int i) const;
unsigned char& operator[](int i);
const irgb_t operator+(const irgb_t& rhs) const;
const irgb_t operator*(const drgb_t& rhs) const;
const irgb_t operator*(const double& s) const;
friend irgb_t operator*(const double& s, const irgb_t& rhs);
friend std::ostream& operator<<(std::ostream& s, const irgb_t& rhs);
friend std::ostream& operator<<(std::ostream& s, irgb_t *rhs);
friend std::istream& operator>>(std::istream& s, irgb_t& rhs);
friend std::istream& operator>>(std::istream& s, irgb_t *rhs);
bool iszero(void) const;
void clamp(unsigned char min, unsigned char max);
```

Forward declarations

```
// pixel.h
template <typename T> class rgb_t;
template <typename T> rgb_t<T> operator*(const T& s, const rgb_t<T>& rhs);
template <typename T> std::ostream& operator<<(std::ostream&, const rgb_t<T>&);
template <typename T> std::ostream& operator<<(std::ostream&, rgb_t<T>*);
template <typename T> std::istream& operator>>(std::istream&, rgb_t<T>&);
template <typename T> std::istream& operator>>(std::istream&, rgb_t<T>*);
```

rgb_t

```
//pixel.h
template <typename T>
class rgb_t
{
  public:
  // constructors (overloaded)
  // copy constructor
  // destructors (default ok)
        ~rgb_t()  { };
  // assignment operator
  // compound assignment operator +=
  // compound assignment operator *=
  // accessor and mutator
  // add this instance's value to rhs
  // multiply this instance's value to rhs
  // multiply the double value to rhs
  // friend operators
 ...
  private:
        T pix[3];
}
```

rgb_t overloaded constructor

```
rgb_t(T ix=0, T iy=0, T iz=0)
{
    pix[0] = ix;
    pix[1] = iy;
    pix[2] = iz;
};
```

```
//main.cpp
rgb_t<double>   dp1(1.0, 2.0, 3.0);
rgb_t<unsigned char> ip1('1', '5', '6');
```

```
dp1.pix[0]=1.0
dp1.pix[1]=2.0
dp1.pix[2]=3.0
```

```
ip1.pix[0]=1
ip1.pix[1]=5
ip1.pix[2]=6
```

rgb_t copy constructor

```
rgb_t(const rgb_t& rhs)
{
   for(int i=0; i<3; i++)
        pix[i] = rhs[i];
}
```

```
//main.cpp
rgb_t<double>  dp1(1.0, 2.0, 3.0);
rgb_t<double>  dp4(dp1);
```

```
dp4.pix[0]=1.0
dp4.pix[1]=2.0
dp4.pix[2]=3.0
```

rgb_t assignment operator

```
rgb_t& operator=(const rgb_t& rhs)
{
   if(this != &rhs)
          for(int i=0; i<3; i++) pix[i] = rhs[i];
   return *this;
}
```

```
//main.cpp
rgb_t<double>  dp1(1.0, 2.0, 3.0);
rgb_t<double>  dp4=dp1;
```

```
dp4.pix[0]=1.0
dp4.pix[1]=2.0
dp4.pix[2]=3.0
```

rgb_t accessors

```
const T& operator[](int i) const
{
      return pix[i];
}
T& operator[](int i) {
      return pix[i];
}
```

```
//main.cpp
 rgb_t<double>  dp2(1.0, 2.0, 3.0);
       dp2[1] = 3.0;
       dp2[0] = dp2[2];
```

```
dp2.pix[0]=3.0
dp3.pix[1]=3.0
dp2.pix[2]=3.0
```

rgb_t operator overloaded functions

```
//pixel.h
rgb_t& operator+=(const rgb_t& rhs){
   if(this != &rhs)
        for(int i=0; i<3; i++)  pix[i] += rhs[i];
   return *this;
 }
 rgb_t& operator*=(const rgb_t& rhs){
      ...
 }
 rgb_t& operator*=(const T& s){
    for(int i=0; i<3; i++) pix[i] *= s;
    return *this;
  }
  const rgb_t operator*(const rgb_t& rhs) const {
    return rgb_t(*this) *= rhs;
  }
...
```

Friend functions

global scope

```
//pixel.h
friend rgb_t (::operator* <>)(const T& s, const rgb_t& rhs);
friend std::ostream& operator<< <>(std::ostream& s, const rgb_t<T>& rhs);
friend std::ostream& operator<< <>(std::ostream& s, rgb_t *rhs);
friend std::istream& operator>> <>(std::istream& s, rgb_t& rhs);
friend std::istream& operator>> <>(std::istream& s, rgb_t *rhs);
```

says there are templated

```
//pixel.h
void clamp(T min, T max)
 {
    for(int i=0;i<3;i++) if(pix[i] < min) pix[i] = min;
    for(int i=0;i<3;i++) if(pix[i] > max) pix[i] = max;
 }
```

```
//pixel.cpp
template <typename T>
bool rgb_t<T>::iszero(void) const
{
   double  precision=0.0000001;
   for(int i=0;i<3;i++)
         if(fabs((double)pix[i]) > precision) return(false);
    return true;
}
```

```
//pixel.cpp
template <typename T>
rgb_t<T> operator*(const T& s, const rgb_t<T>& rhs)
{
  rgb_t<T>    result;
  for(int i=0; i<3; i++) result[i] = s * rhs[i];
  return result;
}
```

friend function or member function?

New pixel.cpp

```
template <typename T>
std::ostream& operator<<(std::ostream& s, const rgb_t<T>& rhs){
       return(s << rhs[0] << " " << rhs[1] << " " << rhs[2]);
}
template <typename T>
std::ostream& operator<<(std::ostream& s, rgb_t<T> *rhs){
      s <<    ?
      return s;
}
```

```
//main.cpp
rgb_t<double> dp1(1.0, 2.0., 3.0);
igb_t<unsigned char> ip1(‘4’, ‘5’, ‘6’);

std::cout << "dp1 = " << dp1 << std::endl;
std::cout<<“ip1=“<<&ip1<<std::endl;
```

```
dp1= 1.0 2.0 3.0

ip1 = 4 5 6
```

New pixel.cpp

```
template <typename T>
std::istream& operator>>(std::istream& s, rgb_t<T>& rhs)
    {
    return(s >> rhs[0] >> rhs[1] >> rhs[2]);
}
template <typename T>
std::istream& operator>>(std::istream& s, rgb_t<T> *rhs){
    s >>          ?
    return s;
}
```

```
//main.cpp
rgb_t<double> dp3;
std::cin>>dp3;

igb_t<unsigned char> ip3;
std::cin>>ip3;
```

Specialization

```
//pixel.cpp
template class rgb_t<double>;
template std::ostream& operator<<(std::ostream&, const rgb_t<double>&);
template std::ostream& operator<<(std::ostream&,       rgb_t<double>*);
template std::istream& operator>>(std::istream&,       rgb_t<double>&);
template std::istream& operator>>(std::istream&,       rgb_t<double>*);

template rgb_t<double> operator*(const double& s, const rgb_t<double>& rhs);

template class rgb_t<unsigned char>;
template std::ostream& operator<<(std::ostream&, const rgb_t<unsigned char>&);
template std::ostream& operator<<(std::ostream&,       rgb_t<unsigned char>*);
template std::istream& operator>>(std::istream&,       rgb_t<unsigned char>&);
template std::istream& operator>>(std::istream&,       rgb_t<unsigned char>*);

template rgb_t<unsigned char> operator*(const unsigned char& s, const rgb_t<unsigned char>& rhs);
```