Operator Overloading

	c1 + c2	==> c1.operator+(c2)
	c1 * c2	==> c1.operator*(c2)
a + ib	c1 == c2	==> c1.operator==(c2)
c + id		

(ac - bd) + i(ad + bc)

Anonymous/Unnamed Object

```
Box b1(10,12,5);
Point p1(3,4);

Box(10,12,5);
Point(3,4)
```

t1 ==> 9:20

Hands-on checklist

Complex number
c1 + c2
c1 - c2
c1 * c2
c1 == c2
c1 + 5

MyTime
t1 + t2
t1 + 25 , t1 + 55
t1 - t2, t1 - 15
t1 == t2

Prev Assignments
Point, Color, MyDate
IPAddress,

Next Focus:-

MyTime:-
++t1
t1++
t1 < t2
t1 > t2
t1 = t2
std::cout << t1
std::cin >> t1

Complex:-
c1==c2
std::cout << c1
std::cin >> c1

Further Assignments
- Fraction class
- Currency class
- Weight, Distance
- MyDate
- Matrix.

tres = t1++
tres = ++t1

a=10
b=a++

a=10
b=++a

t6 = t5 = t1

a = b = c

```
t5 = t1
t6 = t5
```

```
param ctor

default ctor
```

```
MyTime t2 = t1;   //copy ctor
MyTime t3 (t1);   //copy ctor
```

```
copy ctor
```

```
MyTime t4;
```

```
destructor
```

```
t4 = t1;          //operator=
```

```
operator=
```

For trivial classes, overloading assignment operator is not needed
Which will be taken care by compiler

For Non Trivial classes, assignmet operator must be implemented
by used (can be disabled by using =delete), e.g. MyString class

```
Box b1(10,12,5);

Box b2(b1);

Box b3;
b3 = b1;
```

```
class Box
{
  public:
  Box(const Box&)=delete;
  Box& operator=(const Box&)=delete;
}
```

Rule of three/zero
* destructor
* copy ctor
* operator=

Trivial class - No need to implement all above three
(Compiler will take care of)
shouldn't be deleted

Non Trivial class - We need to implement all above three
(or) some may be deleted

	Member Function	Global friend function
t1 + t2	t1.operator+(t2)	operator+(t1,t2)
t1 + 25	t1.operator+(25)	operator+(t1,25)
t1 == t2	t1.operator==(t2)	operator==(t1,t2)
++t1	t1.operator++()	operator++(t1)
t1++	t1.operator++(int)	operator++(t1,int)
t1 < t2	t1.operator<(t2)	operator<(t1,t2)
t1 > t2	t1.operator>(t2)	operator>(t1,t2)
t1 = t2	t1.operator=(t2)	-- Not Possible --
cout << t1	-- Not Possible --	operator<<(cout,t1)
cin >> t1	-- Not Possible --	operator>>(cout,t1)

Note:-
Assignment operator must be implemented as member function only, i.e. can't implemented as friend functions

Further:-

```
std::cout << t1          ==> operator<<(std::cout, t1)
std::cin >> t2           ==> operator>>(std::cin, t1)
```

```
std::cout   : Object of ostream class
std::cin    : Object of istream class

std::cout << x
        ==> operator<<(std::cout, x)
std::cin >> x
        ==> operator>>(std::cin, x)

std::cout << x << y;
```

Hands-on
- Complex
- MyTime
- Fraction class
- Currency / Weight / Distance
- MyDate
- Matrix
- MyString

Unit Testing (GoogleTest)

learncpp.com ==> 21
CppNuts ==> 54, 55, 57, 58, 59

Fraction class
- numerator
- denominator
+ constructors
+ display

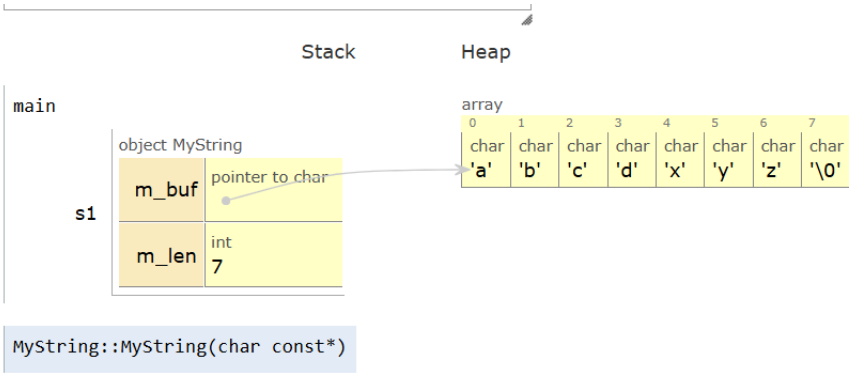
Currency/Weight/Distance
- Rupees / Kilograms / km or meters or feet
- Paisa / grams / m or cm/mm or inches
+ constructors
+ display

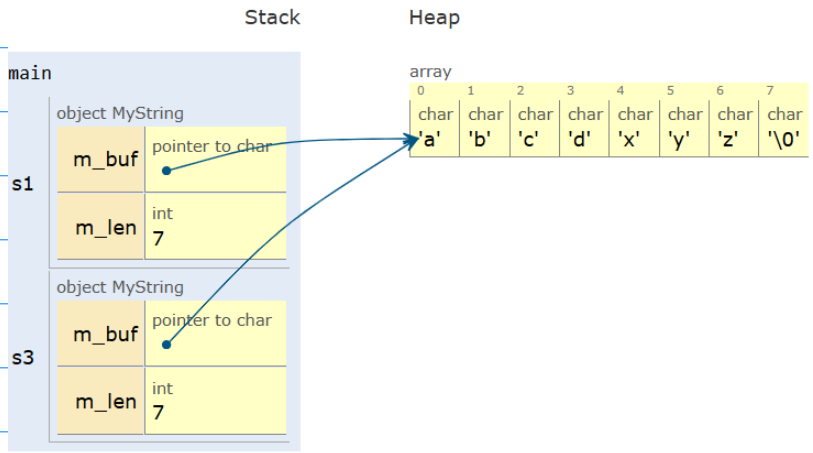
f1 + f2 adding two objects
f1 - f2 subtracting one from other
f1 * f2 adding as scalar
f1 + N subtracting as scalar
f1 - N

==, <, >
f1 == f2 ++ (pre, post)
f1 < f2 operator<<
f1 > f2 operator>>

fres = ++f1
fres = f1++
cout << f1
cin >> f1

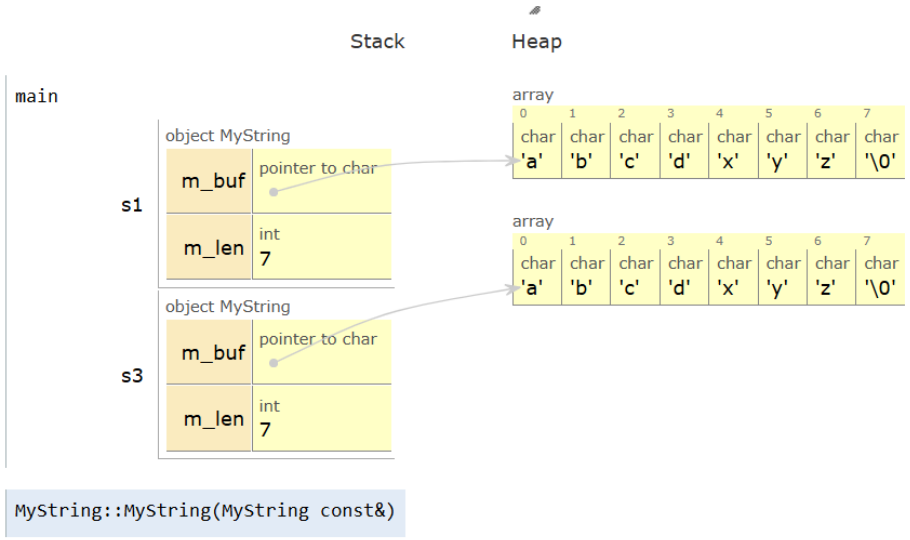
```
std::string s1;
MyString s1("abcdxyz");
```





- changes by one impact other
- one object goes out of scope early (destruct releases memory) m_buf in other object points to invalid location
- When both destroys -- double free prob

Rule of five/zero ==> Move Constructor, Move operator= (Added in C++11)



This is safe, deep copy
- No double free prob
- Changes are independent
- Any one can release heap block independently

Rule of three/zero

	Trivial	Non-Trivial
default ctor		
param ctor	empty body	non-empty body
copy ctor	synthesized	user impl
destructor	do-nothing	must be impl
operator=	synthesized	user impl

synthesized : provided by compiler, member wise copy

Initial C++ by stroustrup : between 80-90
ANSI C++ / C++89 / C++90
C++98 (std C++, by ISO)
C++11 : Modern C++ starts
C++14
C++17
C++20
C++23
C++26

Some C++11 additions

=delete
=default

move ctor
move operator=

Setting up GoogleTest

```
sudo apt install cmake build-essential
git clone https://github.com/google/googletest
cd googletest
mkdir build
cd build
cmake ..
make
sudo make install

#Checks
ls /usr/local/lib       # libgtest.a , libgtest_main.a
ls /usr/local/include   # can see gtest sub folder

g++ factorial.cpp factorial_test.cpp -lgtest -lgtest_main -o factorialdemo
```

leapyear / triangle / prime number

TODO/Hands-on

- Classes & Objects
- Operator Overloading examples
- 1 or 2 gtest example