Algoritmalar ve Programlama II – BLM 102

Hafta 6: Operatör Aşırı Yükleme

Fenerbahçe Üniversitesi

Öğretim Elemanları

Öğretim Üyesi: Dr. Vecdi Emre Levent

Ofis: 311

Email: emre.levent@fbu.edu.tr

Asistan: Arş. Gör. Uğur Özbalkan

Ofis: 311

Email: ugur.ozbalkan@fbu.edu.tr

Ders Planı

- Operatör Aşırı Yükleme

Operatör Aşırı Yükleme

- Sınıflardan yaratılan objelerin operatörler ile işlem yapma yeteneğine kavuşmasını sağlarlar.

- Operatör'ün görevi aşırı yükleme fonksiyonları ile tanımlanmaktadır.

Operatör Aşırı Yükleme

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

class Complex {
private:
int real, imag;
public:
Complex(int r = 0, int i = 0) { real = r;
imag = i; }

Complex operator + (Complex const& obj) {
    Complex res;
    res.real = real + obj.real;
    res.imag = imag + obj.imag;
    return res;
}
```

```
void print() { cout << real << " + i" << imag
<< endl; }
};

int main()
{
Complex c1(10, 5), c2(2, 4);
Complex c3 = c1 + c2;
c3.print();
}
```

```
12 + i9
```

Operatör Aşırı Yükleme

- Özellikle matematiksel işlem (Örn. KarmasikSayilar Sınıfı) barındıran sınıflar için kodlama kolaylığı sağlamaktadır.

- Aşırı yüklenebilecek operatörler:

Operatörler							
+	-	*	/	%	^	&	|
~	!	=	<	>	+=	-=	*=
/=	%=	^=	&=	|=	<<	>>	>>=
<<=	==	!=	<=	>=	&&	||	++
--	->*	,	->	[]	()	new	delete
new[]	delete[]						

Operatör Aşırı Yükleme

Operatör Fonksiyonları

```
Complex operator + (Complex const& obj) {
    Complex res;
    res.real = real + obj.real;
    res.imag = imag + obj.imag;
    return res;
}
```

Operatör Fonksiyonu

- Diğer fonksiyonlardan bir farkı yoktur, sadece başında "operatör" keyword'ü bulunmaktadır.

Operatör Aşırı Yükleme

Operatör Fonksiyonları

- Bu fonksiyonlar, operatörlerin görevlerini tanımlamaktadırlar.

Operatör Aşırı Yükleme

Operatör Fonksiyonları

- Bir sınıfa ait olmayan global fonksiyonlar'da operatör aşırı yükleme gerçekleştirebilir.

Operatör Aşırı Yükleme

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

class Complex {
private:
int real, imag;
public:
Complex(int r = 0, int i = 0) { real = r;   imag = i;
}
void print() { cout << real << " + i" << imag <<
endl; }

friend Complex operator + (Complex const&, Complex
const&);
friend Complex operator + (Complex const&, int);
};
```

```
Complex operator + (Complex const& c1, Complex const&
c2)
{
return Complex(c1.real + c2.real, c1.imag + c2.imag);
}

Complex operator + (Complex const& c1, int a)
{
return Complex(c1.real + a, c1.imag + a);
}


int main()
{
Complex c1(10, 5), c2(2, 4);
Complex c3 = c1 + c2;
c3.print();
Complex c4 = c1 + 5;
c4.print();
return 0;
}
```

Operatör Aşırı Yükleme

Operatör aşırı yükleme yapmak için;

- Operandlardan (İşleme giren sayılardan) en az birisi kullanıcının tanımladığı bir sınıf olmalıdır.

Operatör Aşırı Yükleme

Operatör aşırı yükleme yapmak için;

- Bir sınıftan bir veri türüne dönüşümü operatör aşırı yükleme ile gerçekleştirilebilir.

Operatör Aşırı Yükleme

```
#include <iostream>
using namespace std;
class Fraction
{
int num, den;
public:
Fraction(int n, int d) { num = n; den = d; }

operator float() const {
return float(num) / float(den);
}
};

int main() {
Fraction f(2, 5);
float val = f;
cout << val;
return 0;
}
```

Operatör Aşırı Yükleme

Tekli operatör aşırı yükleme

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

class Distance {
private:
int feet;
int inches;

public:
Distance() {
     feet = 0;
     inches = 0;
}
Distance(int f, int i) {
     feet = f;
     inches = i;
}
void displayDistance() {
          cout << "F: " << feet << " I:" << inches << endl;
}

Distance operator- () {
     feet = -feet;
     inches = -inches;
     return Distance(feet, inches);
}
};

int main() {
Distance D1(11, 10), D2(-5, 11);

-D1;
D1.displayDistance();

-D2;
D2.displayDistance();

return 0;
}
```

Operatör Aşırı Yükleme

Giriş/Çıkış Aşırı Yükleme

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

class Distance {
private:
int feet;
int inches;

public:
Distance() {
feet = 0;
inches = 0;
}
Distance(int f, int i) {
feet = f;
inches = i;
}
friend ostream& operator<<(ostream& output, const Distance&
D) {
output << "F : " << D.feet << " I : " << D.inches;
return output;
}

friend istream& operator>>(istream& input, Distance& D) {
input >> D.feet >> D.inches;
return input;
}
};

int main() {
Distance D1(11, 10), D2(5, 11), D3;

cout << "Enter the value of object : " << endl;
cin >> D3;
cout << "First Distance : " << D1 << endl;
cout << "Second Distance :" << D2 << endl;
cout << "Third Distance :" << D3 << endl;
return 0;
}
```

```
Enter the value of object :
70
10
First Distance : F : 11 I : 10
Second Distance :F : 5 I : 11
Third Distance :F : 70 I : 10
```