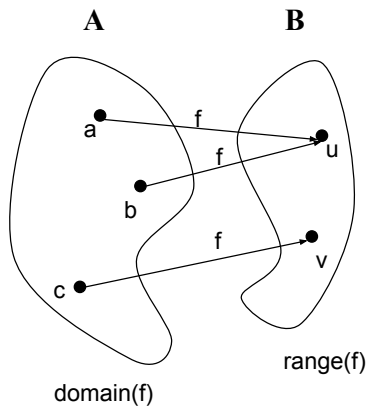


# Представяне на функции и операции с тях

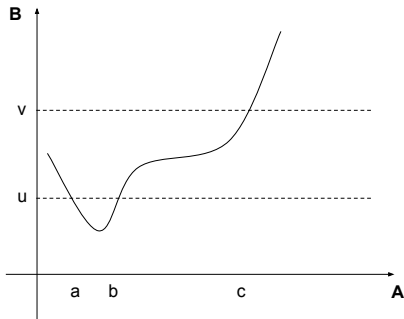
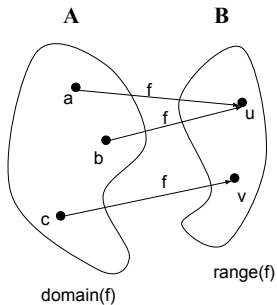
Калин Георгиев

21 май 2020 г.

Какво е функция?

Изображение  $f : A \rightarrow B$ 

Графика на функцията,  $G(f) = \{(x, f(x)) | x \in \text{range}(f)\}$



Как представяме функция компютърно?

# Таблично представяне

x	f(x)
a	u
b	u
c	v

# Представяне чрез програма

```
double f (double x)
{
    if (x == a) return u;
    if (x == b) return u;
    if (x == c) return v;
    return 0;
}
```

Представяне на функцията чрез “атрибути”,  
които я дефинират еднозначно



# Константна функция

$$f(x) = 5$$

```
class Constant
{
    private:
        double c;
    public:
        Constant (double val): c(val){}
}
```

# Линейна функция

$$f(x) = 2x + 5$$

```
class Linear
{
    private:
        double coef;
        double constant;
    public:
        Linear (double a, double b):
            coef(a), constant(b){}
}
```

# Полином

$$f(x) = a_0x^n + a_1x^{n-1} + \dots + a_n$$

```
class Polynom
{
    private:
        vector<double> coefs;
    public:
        Polynom (vector<double>arr):
            coefs(arr){}
}
```

Кое е общото между всички  
едноместни числови функции?

$$y = f(x)$$

```
class Function
{
    public:
    virtual double value (double x) = 0;
}
```

# Константна функция

$$f(x) = 5$$

```
class Constant : public Funtion
{
    private:
        double c;
    public:
        Constant (double val):c(val){}

        double value (double x) {return c};
}
```

# Линейна функция

$$f(x) = 2x + 5$$

```
class Linear : public Funtion
{
    private:
        double coef;
        double constant;
    public:
        Linear (double a, double b):
            coef(a), constant(b){}

        double value (double x)
        {
            retuen coef*x + constant;
        }
}
```

# Полином

$$f(x) = a_0x^n + a_1x^{n-1} + \dots + a_n$$

```
class Polynom : public Funtion
{
    ...
    double value (double x)
    {
        double sum = coefs[0];
        for (int i = 1; i < coefs.size()-1; i++)
            sum = sum * x + coefs[i];
        return sum;
    }
}
```



## Оператори над функции

# Функция vs. Оператор

- Функция:  $f : A \rightarrow B, g : B \rightarrow C$
- Оператор:  $\Gamma : (A \rightarrow B) \rightarrow (A \rightarrow B)$
- Оператор:  $\Gamma : (A \rightarrow B) \times (B \rightarrow C) \rightarrow (A \rightarrow C)$

$$\Gamma : (double \rightarrow double) \rightarrow (double \rightarrow double)$$

$$\Gamma(f)(x) = \begin{cases} f(x) & f(x) \geq 0 \\ 0 & otherwise \end{cases}$$

```
class CutFunction : public Function
{
private:
    Function* f;
public:
    CutFunction (Function *_f):f(_f){}
    double value (double x){
        double y = f->value(x);
        if (y >= 0)
            return y;
        return 0;
    }
}
```

# Използване на CutFunction

```

class CutFunction : public Function
{
private:
    Function* f;
public:
    CutFunction (Function *_f):f(_f){}
    double value (double x){
        double y = f->value(x);
        if (y >= 0)
            return y;
        return 0;
    }
}

int main ()
{
    Linear fn (-2,10);
    CutFunction cfn (&fn);

    cout << fn.value (10)
         << endl
         << cfn.value (10);
}

```

## $f$ и $CutFn(f)$ са функции

```

void printall (Function *functions[],
               int n,
               double x)
{
    for (int i = 0; i < n; i++)
        cout << functions[i]->value(x) << endl;
}

int main ()
{
    Linear fn (-2,10);
    CutFunction cfn (&fn);
    Function *functions[] = {&fn,&cfn};
    printall (functions,2,10);
}

```

$$\Gamma : (\text{double} \rightarrow \text{double}) \times (\text{double} \rightarrow \text{double}) \rightarrow (\text{double} \rightarrow \text{double})$$

$$\Gamma(f, g)(x) = f(g(x))$$

```
class Composition : public Function
{
    private:
        Function* f;
        Function* g;
    public:
        Composition (Function *_f, Function *_g)
            :f(_f),g(_g){}
        double value (double x){
            return f->value (g->value (x));
        }
}
```

# Използване на Composition

```
int main ()
{
    Linear fn (-2,10);
    CutFunction cfn (&fn);
    Composition comp (&fn,&cfn);

    cout << comp.value (10);

    Function *functions[] = {&fn,&cfn,&comp};
    printall (functions,3,10);
}
```

## Програмите като оператори



# IF като оператор

```
double G (double x)
{
    if (f(x) != 0)
        return g(x);
    return h(x);
}
```

$$\Gamma(f, g, h)(x) = \begin{cases} g(x) & f(x) \neq 0 \\ h(x) & \text{otherwise} \end{cases}$$

# IF като оператор

$$\Gamma(f, g, h)(x) = \begin{cases} g(x) & f(x) \neq 0 \\ h(x) & \text{otherwise} \end{cases}$$

```
class IfOperator : public Function
{
private:
    Function* condfn; //f
    Function* thenfn; //g
    Function* elsefn; //h
public:
    IfOperator (Function *_c,
                Function *_t,
                Function *_e): condfn(_c),
                                thenfn(_t),
                                elsefn(_e){}
}
```

# IF като оператор

$$\Gamma(f, g, h)(x) = \begin{cases} g(x) & f(x) \neq 0 \\ h(x) & \text{otherwise} \end{cases}$$

```
class IfOperator : public Function
{
    public:
        double value (double x){
            if (condfn->value(x) == 0)
                return elsefn->value (x);
            return thenfn->value (x);
        }
}
```

# FOR като ператор

```
double G (double x)
{
    double z = x;
    for (int i = 0; i < N; i++)
        z = f(z);
    return z;
}
```

$$\Gamma(f, n)(x) = \underbrace{f(f \dots (f(x) \dots))}_n$$

# FOR като оператор

$$\Gamma(f, n)(x) = \underbrace{f(f \dots (f(x) \dots))}_n$$

```
class ForOperator : public Function
{
private:
    Function* fn;
    int count;
public:
    ForOperator (Function *_fn):fn(_fn){}
    double value (double x)
    {
        double z = x;
        for (int i = 0; i < count; i++)
            z = fn->value(z);
        return z;
    }
}
```

Благодаря ви за вниманието!