

**Rīgas Tehniskā universitāte
Datorzinātnes un informācijas tehnoloģijas fakultāte
Lietišķo datorsistēmu institūts
Lietišķo datorzinātņu katedra**

DPI230 Objektorientētā programmēšana

Profesors Uldis Sukovskis

Mērķis un uzdevumi

- n** Izprast un spēt izskaidrot objektorientētās programmēšanas principus un tehnoloģijas, spēt tās lietot programmatūras izstrādes procesā, izmantojot objektorientēto programmēšanas valodu vizuālās programmēšanas vidēs.
- n** Studentiem patstāvīgi jāstrādā ar literatūras avotiem, patstāvīgi jāsagatavo un jāizpilda individuālie laboratorijas darbu uzdevumi.

Sasniedzamie studiju rezultāti

- n Spēj izskaidrot OO pamatjēdzienus un demonstrēt to realizāciju valodā C++.
- n Spēj patstāvīgi realizēt programmas valodā C++, izmantojot objektorientētās programmēšanas tehnoloģiju, izskaidrot to darbību un izmantotos operatorus.
- n Spēj analizēt un izskaidrot C++ pirmkodu un izskaidrot tajā izmantotos operatorus.
- n Spēj izskaidrot objektorientētās un strukturālās programmēšanas īpašības.

Vērtēšana

n Laboratorijas darbi

- § Laboratorijas darbi jāizpilda tikai Lietišķo datorzinātņu katedras datorklasē
- § Atzīme par laboratorijas darbiem ietilpst gala atzīmē

n Eksāmens

- § Kursa beigās jākārtu rakstisks eksāmens
- § Gala atzīmi (A) aprēķina no eksāmena atzīmes (E) un atzīmes par laboratorijas darbiem (LD)

$$A = 0.65 * E + 0.35 * LD, \text{ ja } E > 3$$

$$A = E, \text{ ja } E < 4$$

Mācību materiāli

n Mācību materiāli tiek ievietoti portāla ORTUS e-studiju vidē

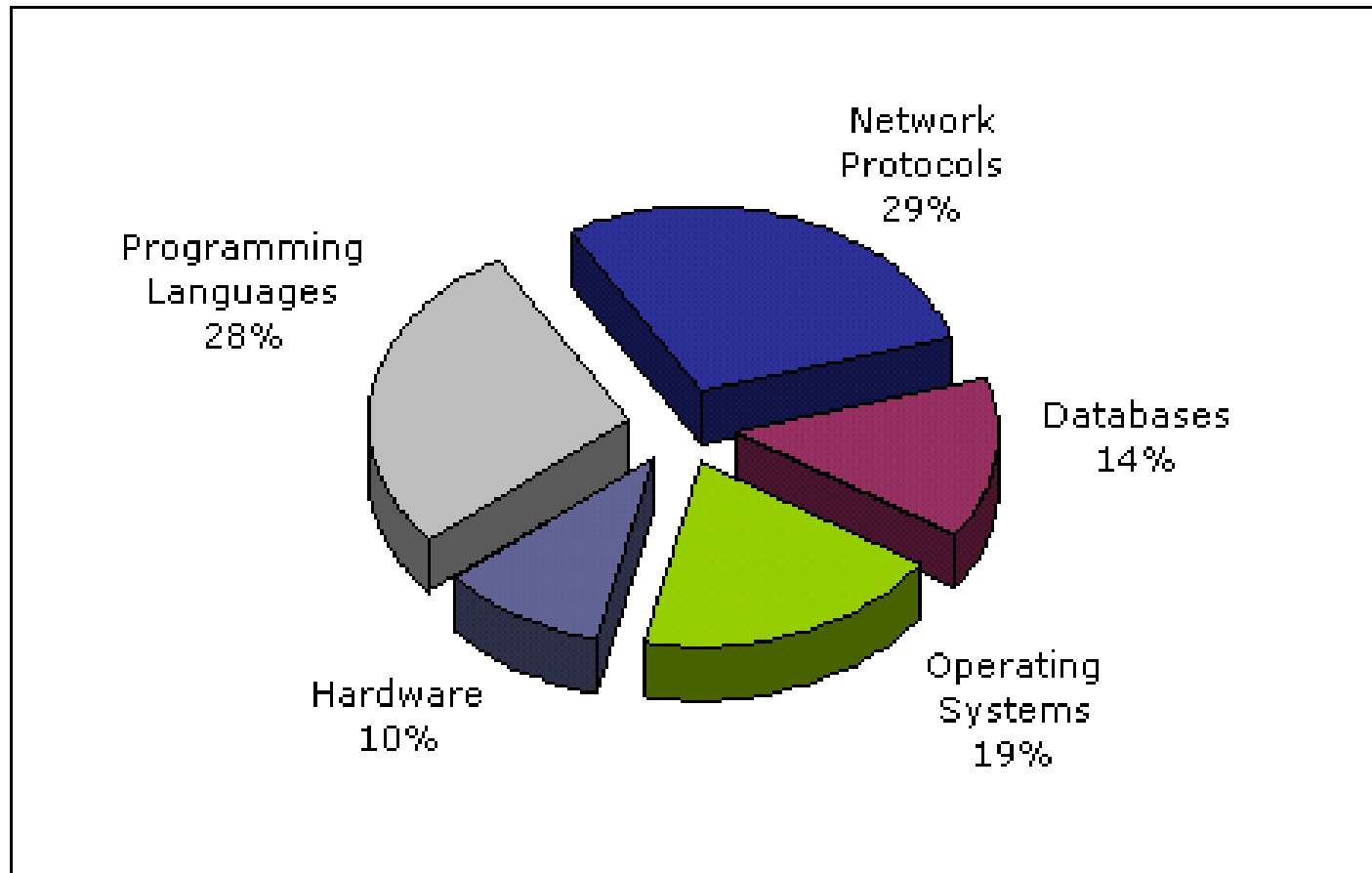
- § Prezentāciju slaidi
- § Programmu piemēri
- § Citi materiāli

n Noderīgas saites par objektorientēto programmēšanu C++

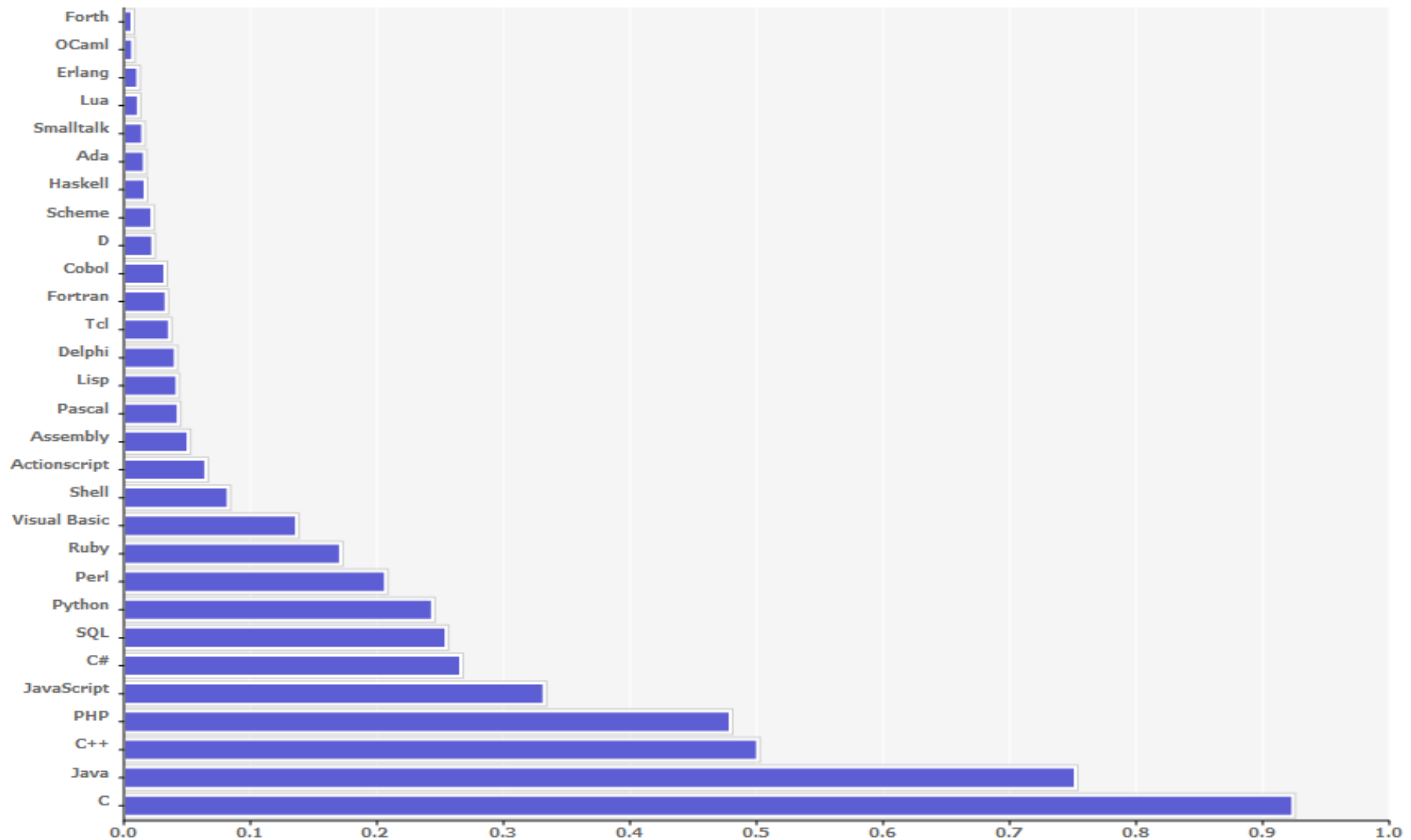
- § www.cplusplus.com
- § www.cpp-tutorial.com/tutorial-cpp.html
- § www.cprogramming.com
- § www.programmingtutorials.com
- § cplus.about.com
- § www.thefreecountry.com/compilers/cpp.shtml

IT zināšanu pieprasījums darba tirgū

Which skill-set was in highest demand in IT projects?



Programmēšanas valodu popularitāte

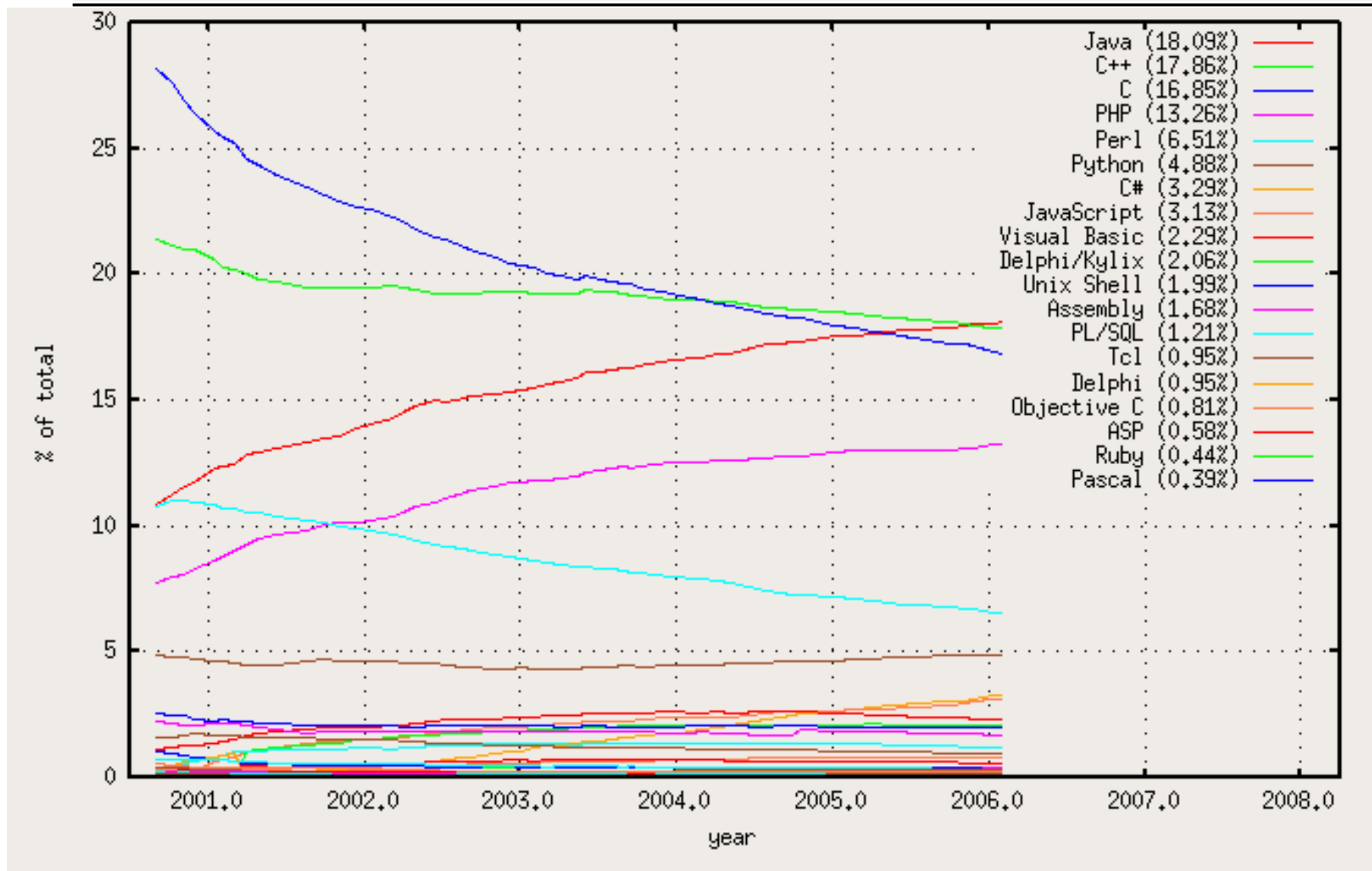


Avots: <http://www.langpop.com/>



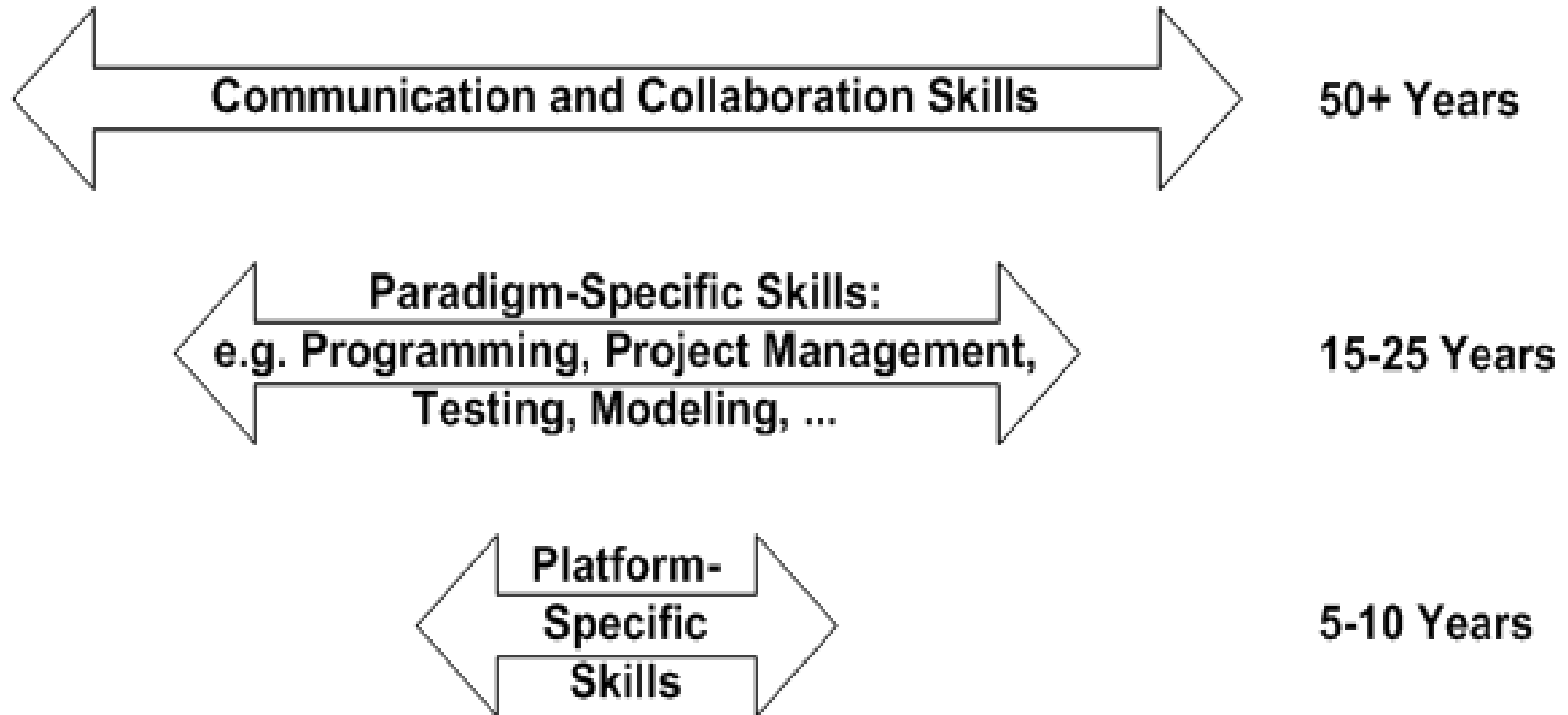
Valodu lietojuma statistika

(atvērtā koda programmatūra)



Avots: <http://www.cs.berkeley.edu/>

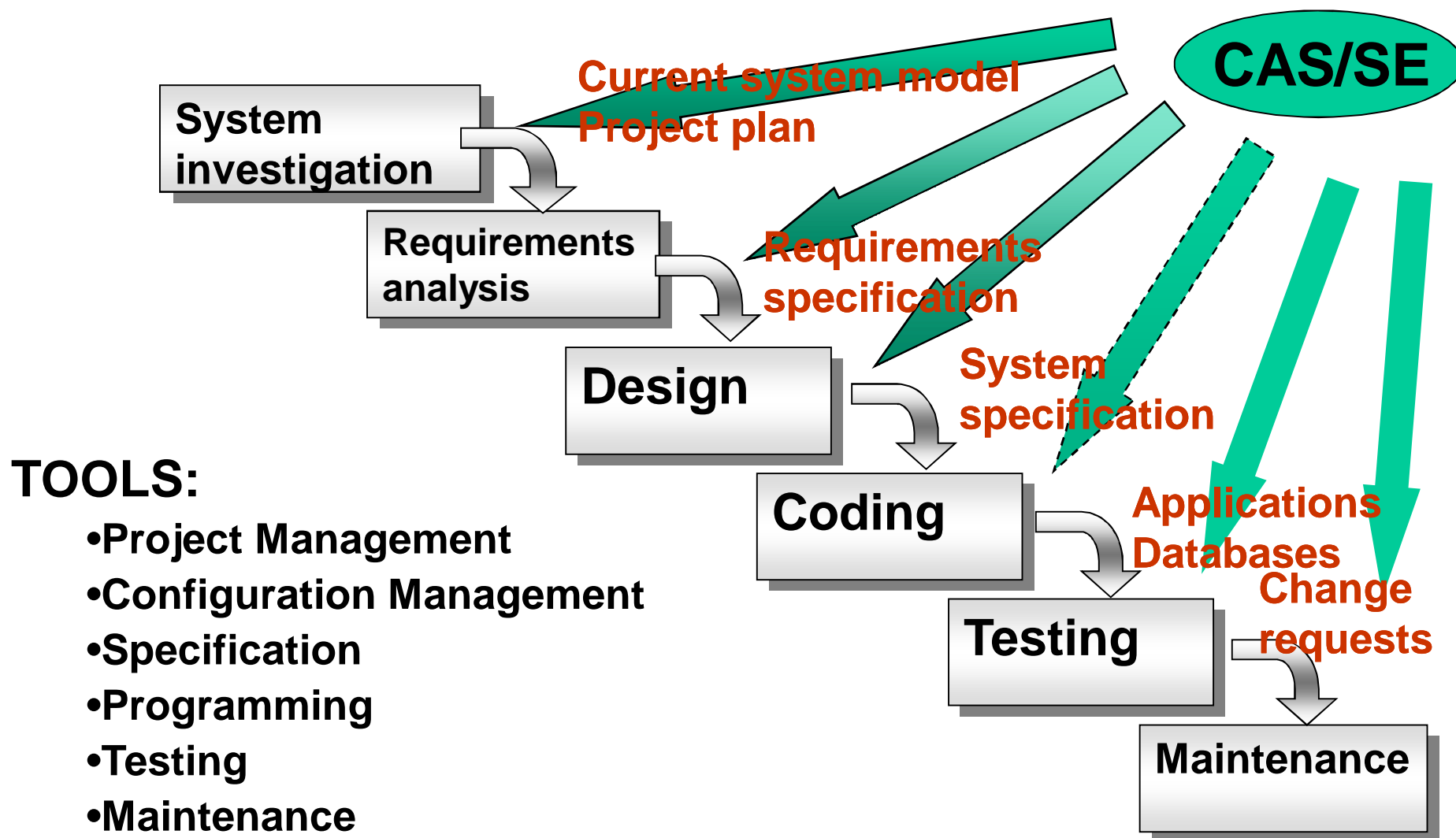
IT zināšanu ilglaicīgums



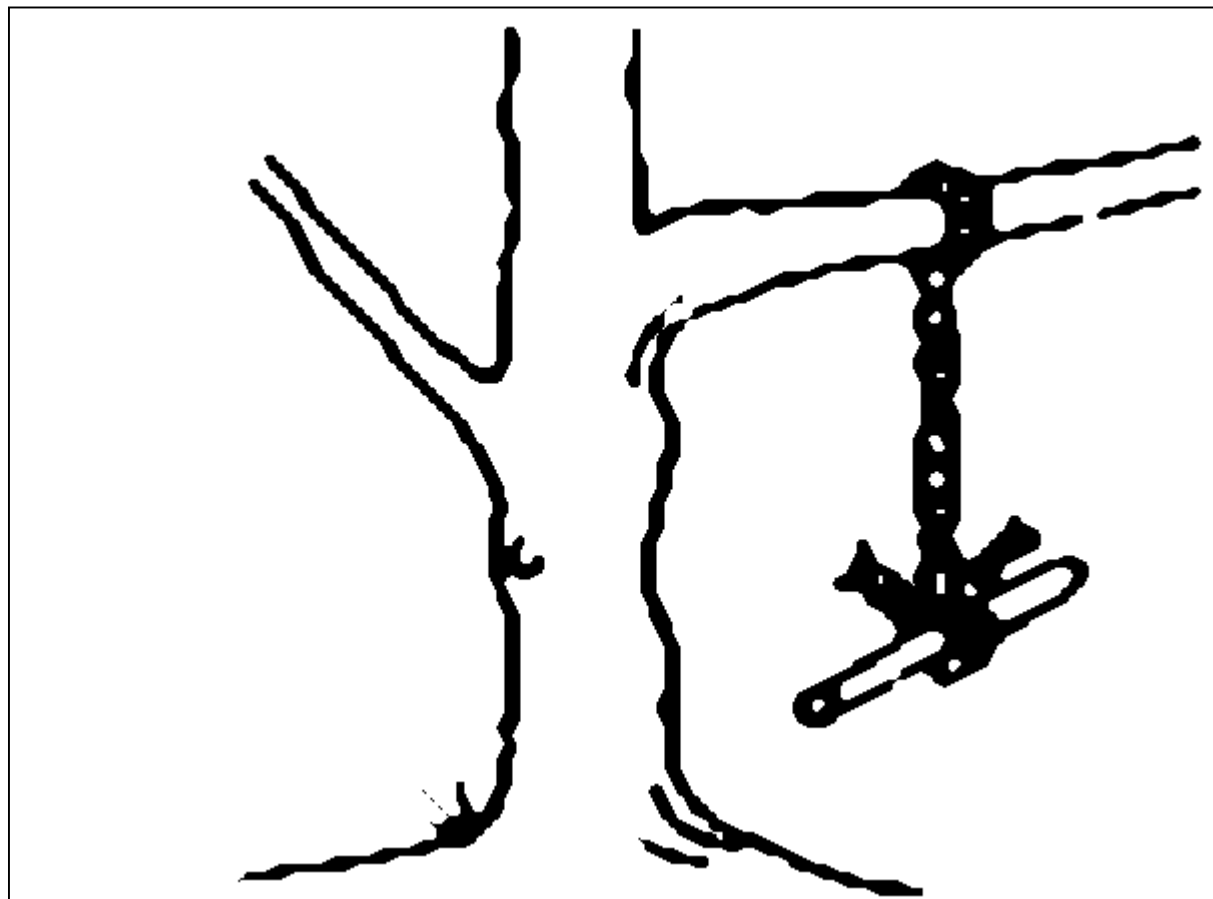
Avots: Scott W. Ambler

Programmatūras izstrādes dzīves cikls

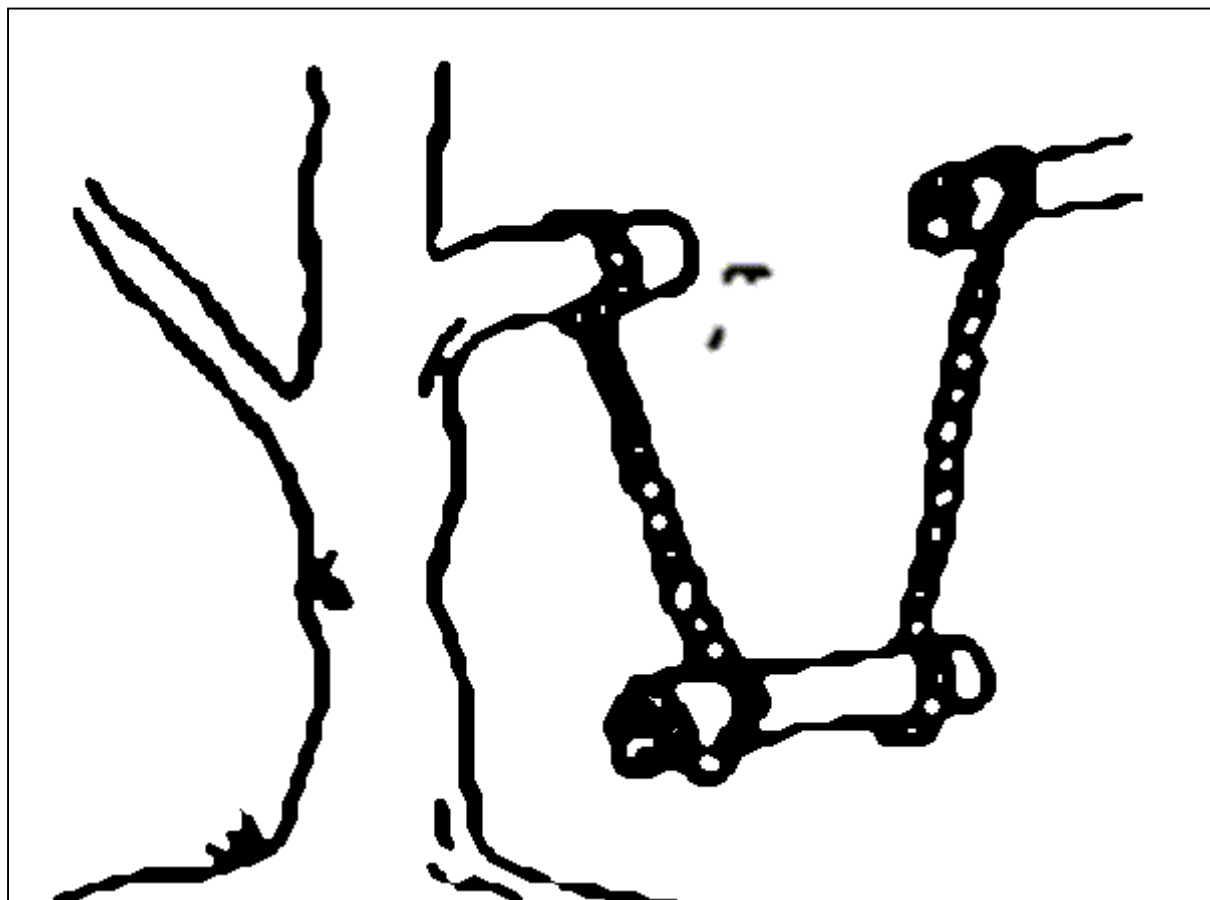
Software Development Life Cycle (SDLC)



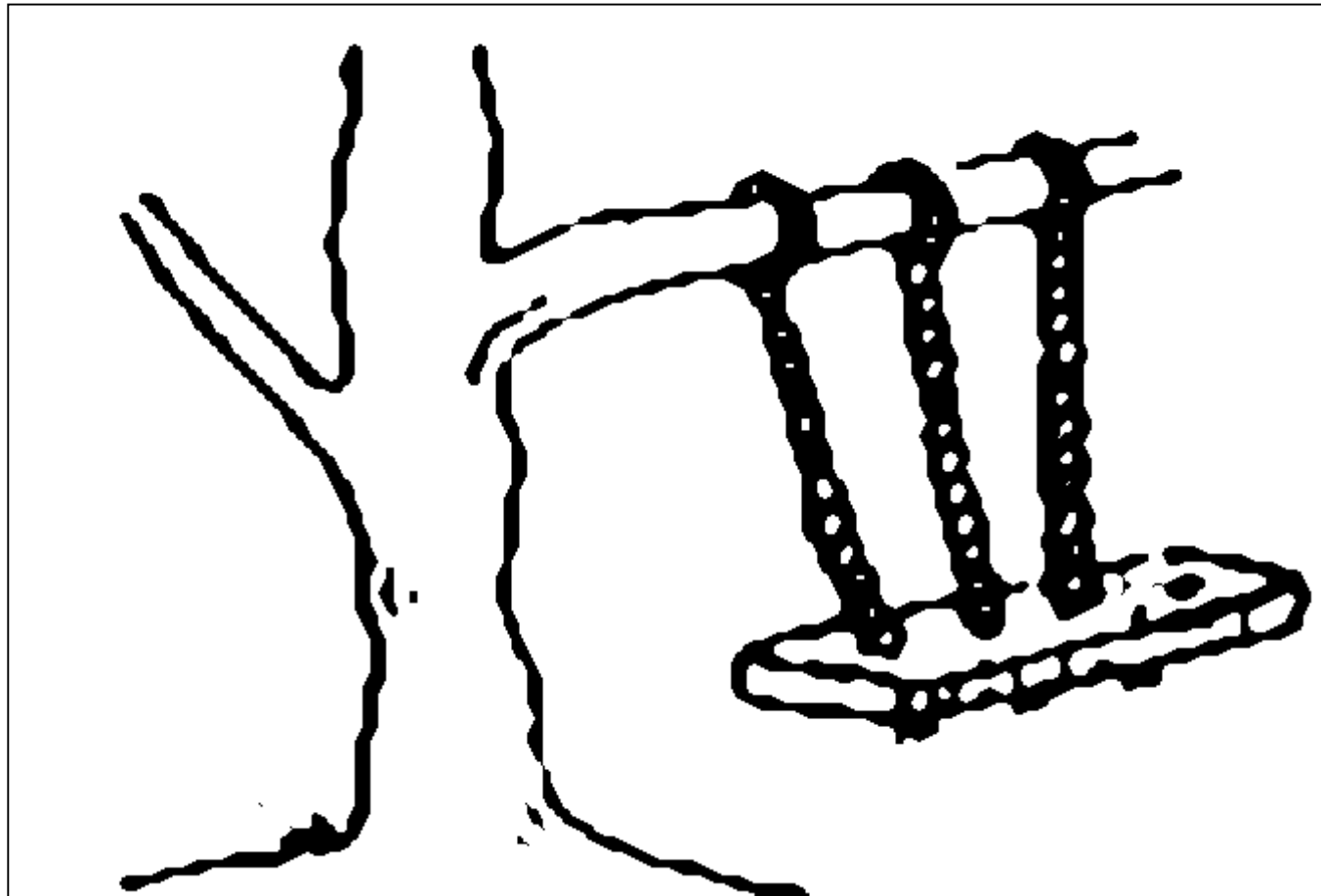
What the user asked for ?



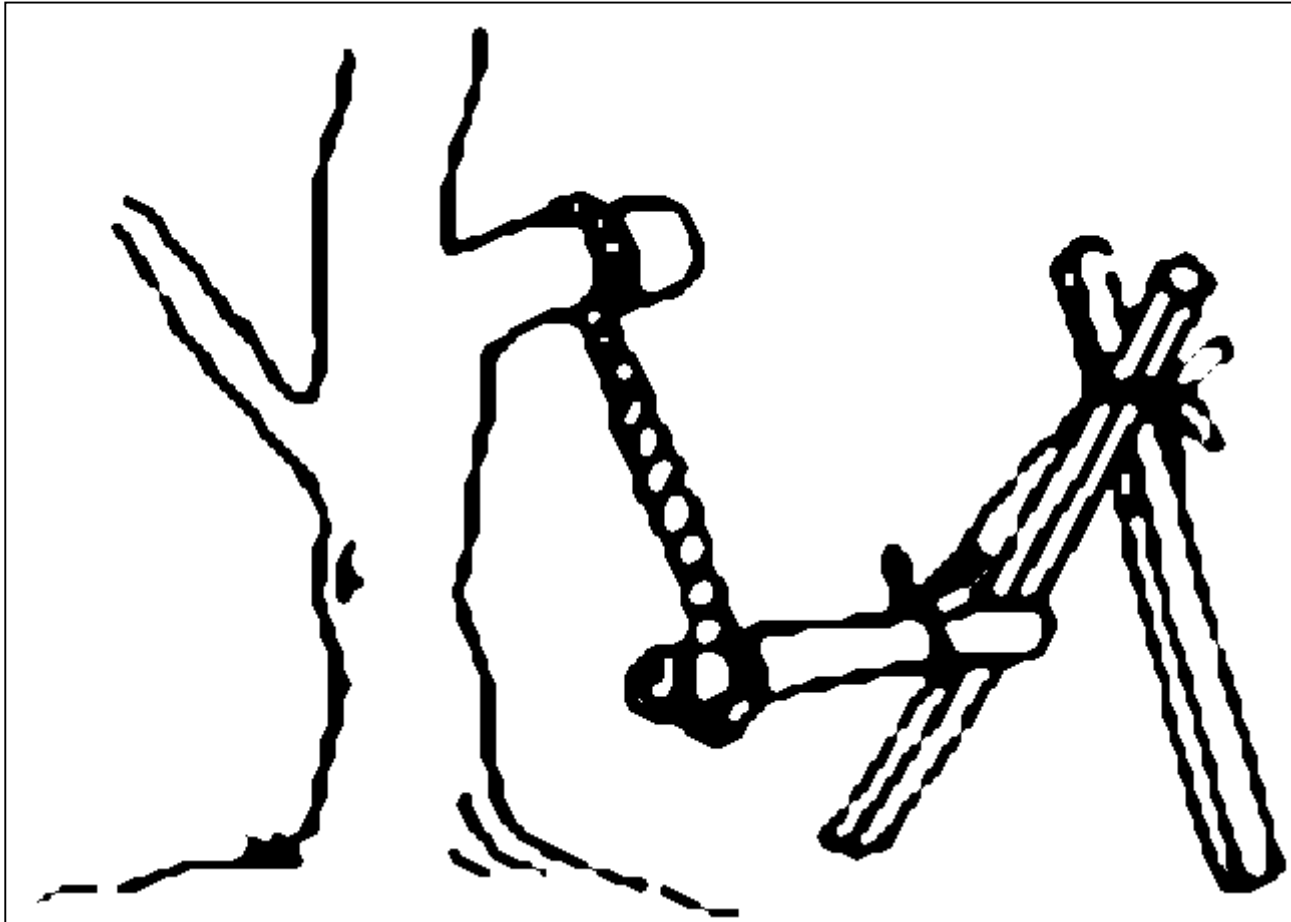
How the analyst saw it ?



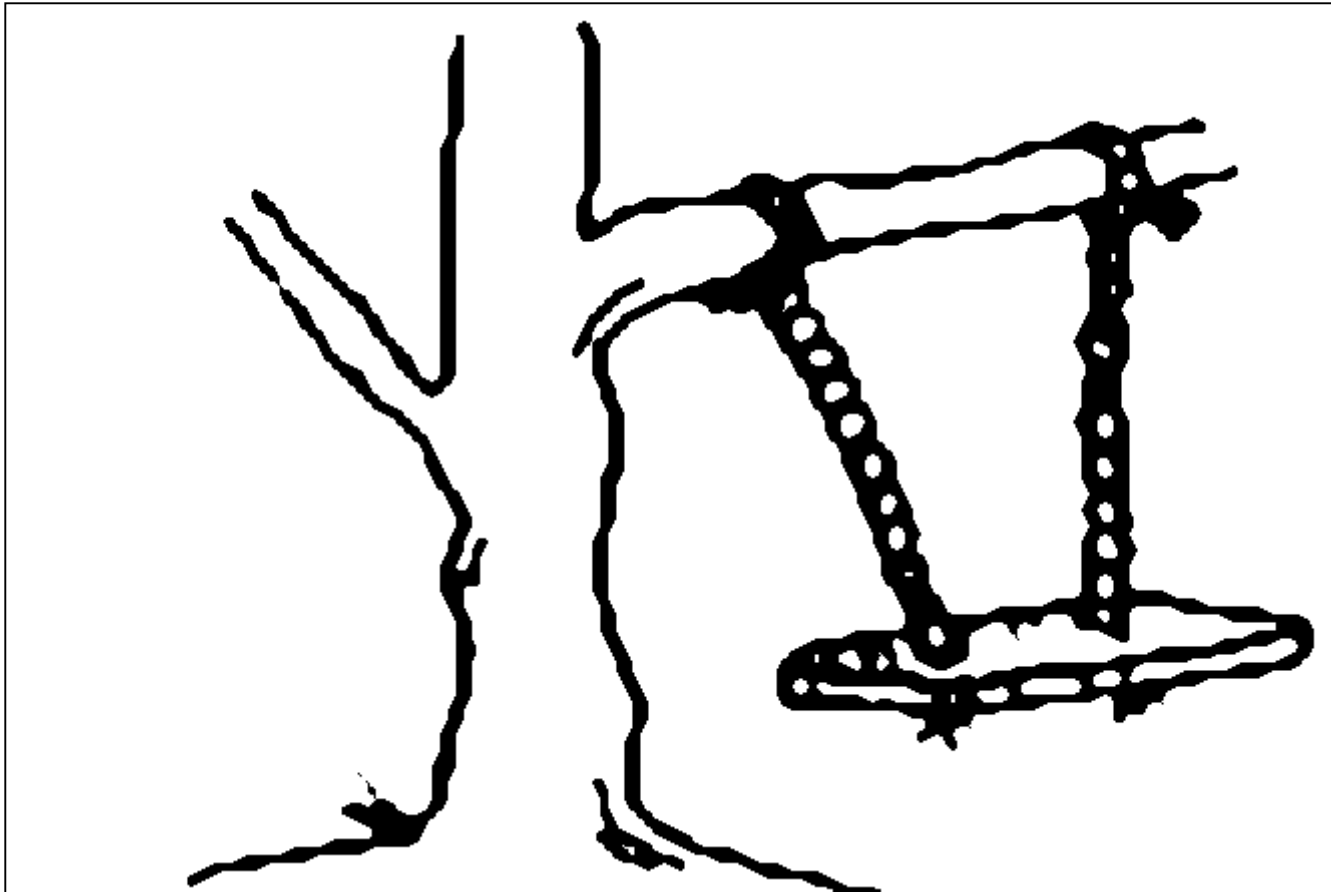
How the system was designed?



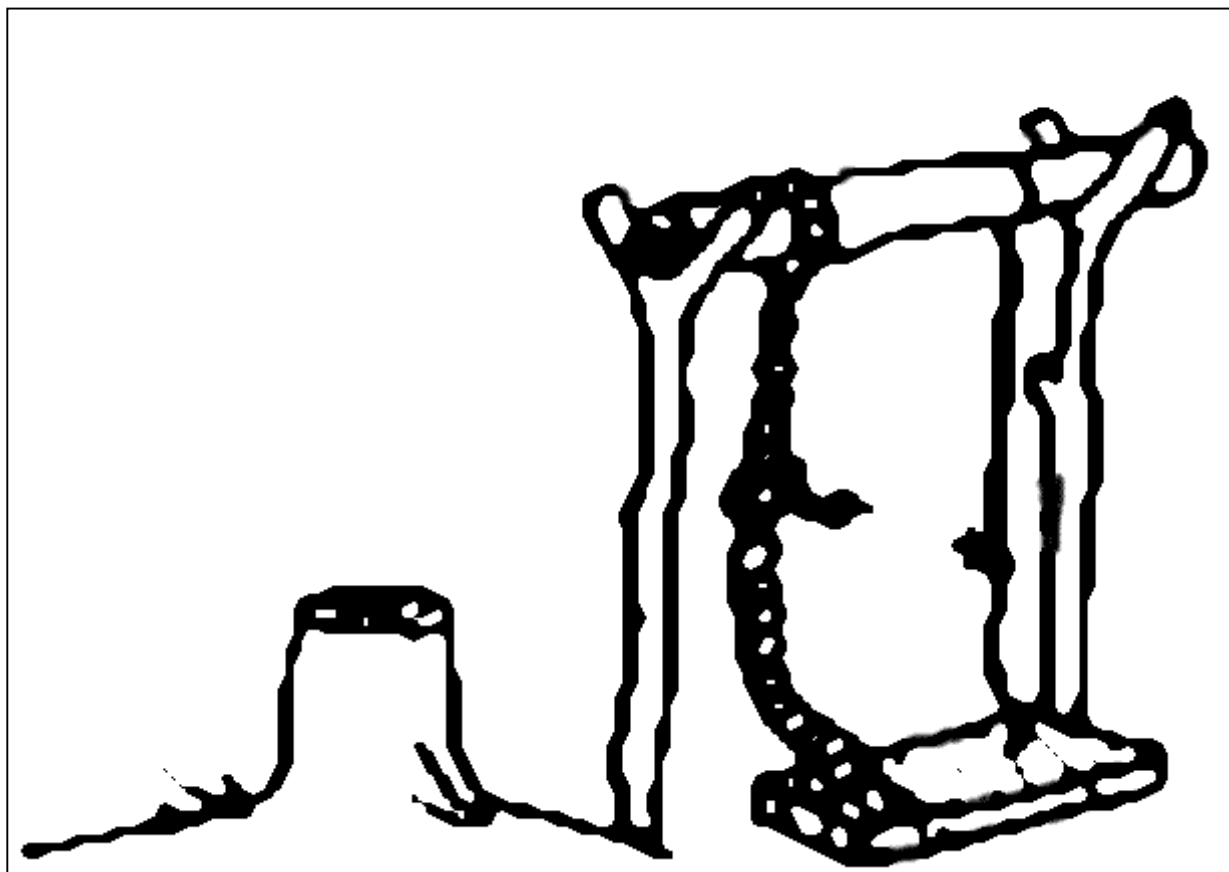
As the programmer wrote it



What the user really wanted?



How it actually works?



Vienkāršas programmas piemērs

Uzrakstīt programmu, kas ļauj aprēķināt trīsstūra laukumu, ja doti tā malu garumi.

Lietosim Hērona formulu :

$$S = \sqrt{p(p-a)(p-b)(p-c)}$$

a, b, c - trīsstūra malu garumi,

p - pusperimetrs

$$p = \frac{a+b+c}{2}$$

Trīsstūra klase

[Triangle.h](#)

```
class Triangle {  
    public:  
        int a, b, c;  
        float area();  
        int perimeter();  
};
```

Trīsstūra klases funkcijas

[Triangle.cpp](#)

```
#include <math.h>
#include "Triangle.h"

float Triangle::area()
{
    float p;
    p = (a + b + c)/2.0; // half perimeter
    return (sqrt(p*(p - a)*(p-b)*(p - c)));
}

int Triangle::perimeter()
{
    return (a + b + c);
}
```

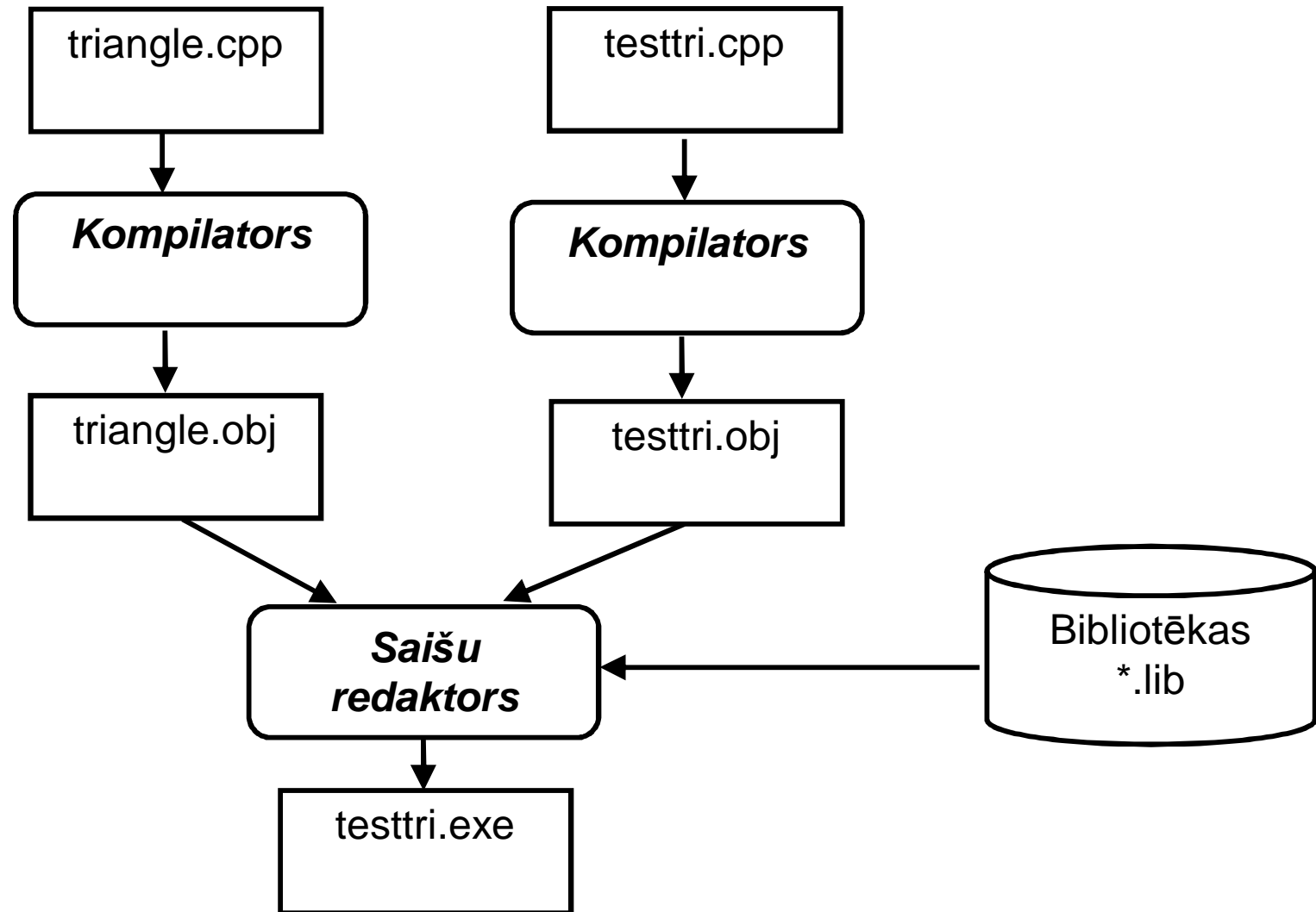
Kā tas strādā?

Testtri.cpp

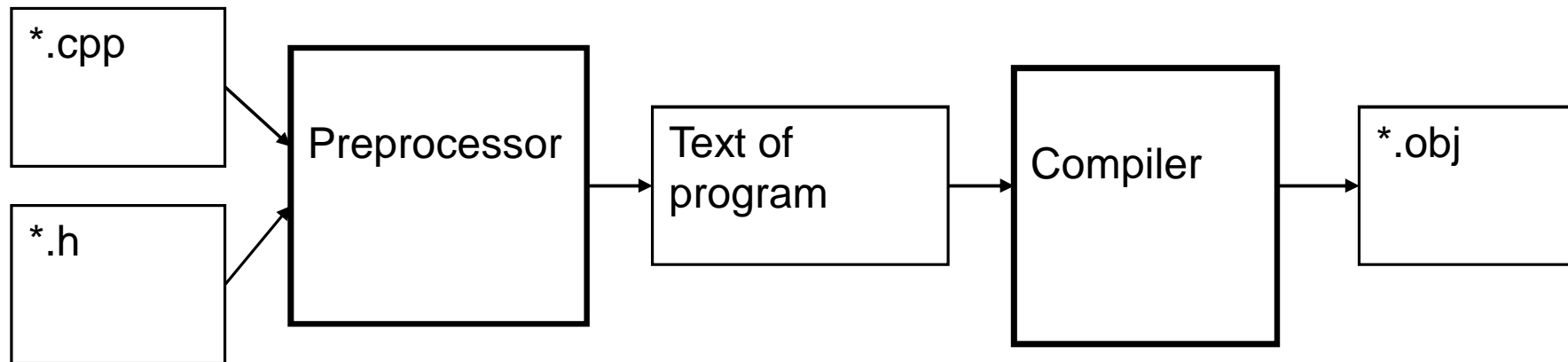
```
#include <iostream.h>
#include "Triangle.h"

void main()
{
    Triangle t;    // Objekts
    t.a = 50;      // Malu garumi
    t.b = 60;
    t.c = 35;
    float ta;
    ta = t.area();
    cout<< "Area of triangle is " << ta << '\n';
    cout<< "Perimeter is "<<t.perimeter()<< '\n';
}
```

Programmas izveidošana



Preprocessors



Programmas tekstā rindas, kas sākas ar simbolu #, tiek apstrādātas kā preprocesora direktīvas.

`#define`

`#undef`

`#ifdef`

`#ifndef`

`#include`

Preprocessors (turpinājums)

#define *identifikators* [*virkne*]

```
#define PI 3.14159
```

```
• • •  
c = 2 * PI * r;      // c = 2 * 3.14159 * r;  
cout << "2*PI*r";    // cout << "2*PI*r";
```

```
#define SIZE 128
```

```
• • •  
#undef SIZE
```

```
• • •  
#define SIZE 1024
```

```
• • •  
#undef SIZE
```

```
#ifndef SIZE
```

```
#define SIZE 1024
```

```
#endif
```


Preprocessors (turpinājums)

Fails MyClass.h

```
#ifndef _MYCLASS_H_
#define _MYCLASS_H_

class MyClass {
    int Duplicate(int x);
};
#endif _MYCLASS_H_
```

Fails MyClass.cpp

```
#include "MyClass.h"

int MyClass::Duplicate(int x) {
    return x + x;
}
```

Fails Main.cpp

```
#include "MyClass.h"

void main(void) {
    MyClass c1;
    ...
}
```

Preprocessors (turpinājums)

`#define identifikators(parametri) virkne`

```
#define CUBE(X) X * X * X
```

```
. . .
```

```
int a, b;
```

```
. . .
```

```
a = 2;
```

```
b = CUBE(a);
```

```
    // b = a * a * a;
```

```
b = CUBE(a+1);
```

```
    // b = a + 1 * a + 1 * a + 1;
```

```
#define CUBE(X) ((X) * (X) * (X))
```

```
. . .
```

```
b = CUBE(a+1);
```

```
    // b = ((a + 1) * (a + 1) * (a + 1));
```

Preprocessors (turpinājums)

```
#define CUBE(X) ((X) * (X) * (X))
```

```
int cube(int x)
{
    return x * x * x;
}
```

```
int a, b;
```

```
a = 3;
```

```
b = cube( a++ );    // a = 4, b = 27
```

```
a = 3;
```

```
b = CUBE( a++ );    // a = 6, b = 27 !!!
```

Preprocessors (turpinājums)

`#include < file >`

`#include " file "`

`#include name`

`#include <string.h>` no standarta kataloga

`#include "mystring.h"` no tekošā kataloga

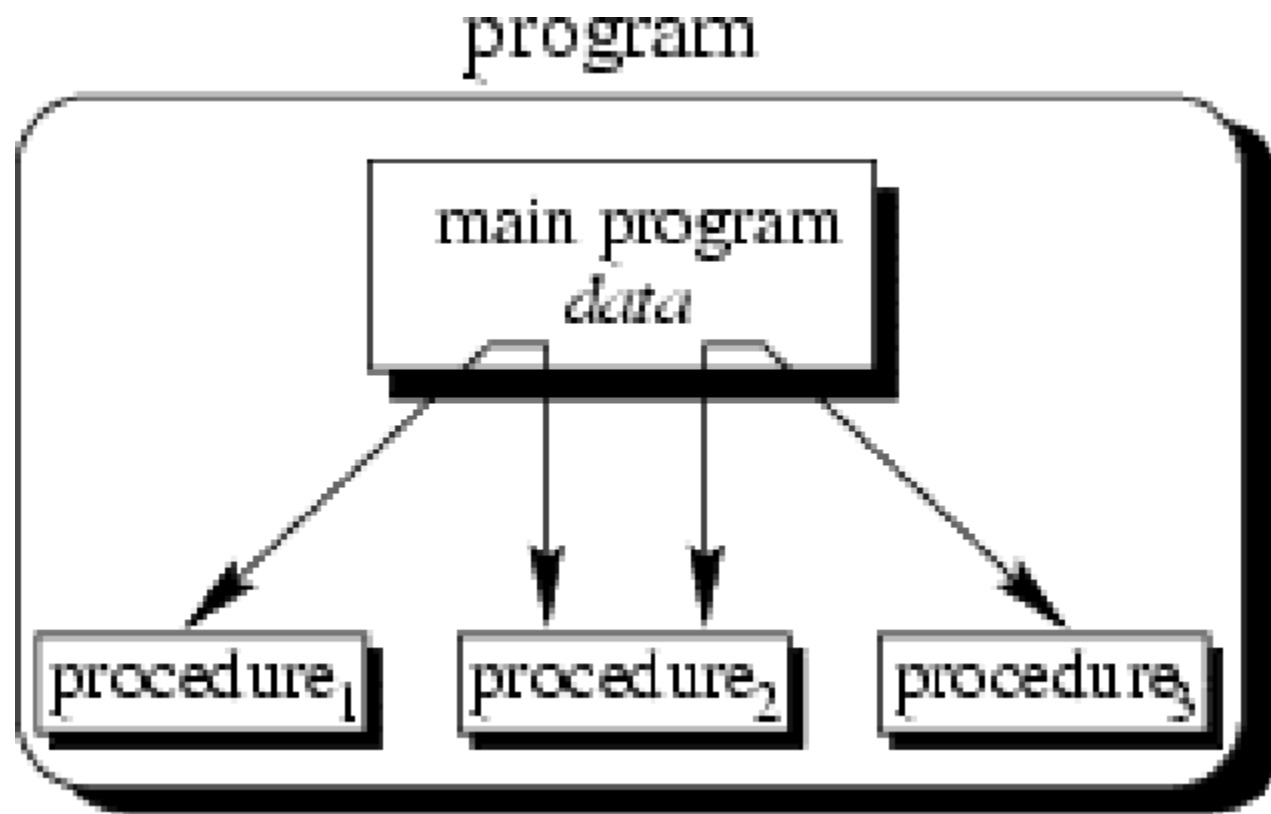
`#include "c:\cprog\sample25\tryit.h"`

`#define MYHEADER "c:\cprog\sample25\second.h"`

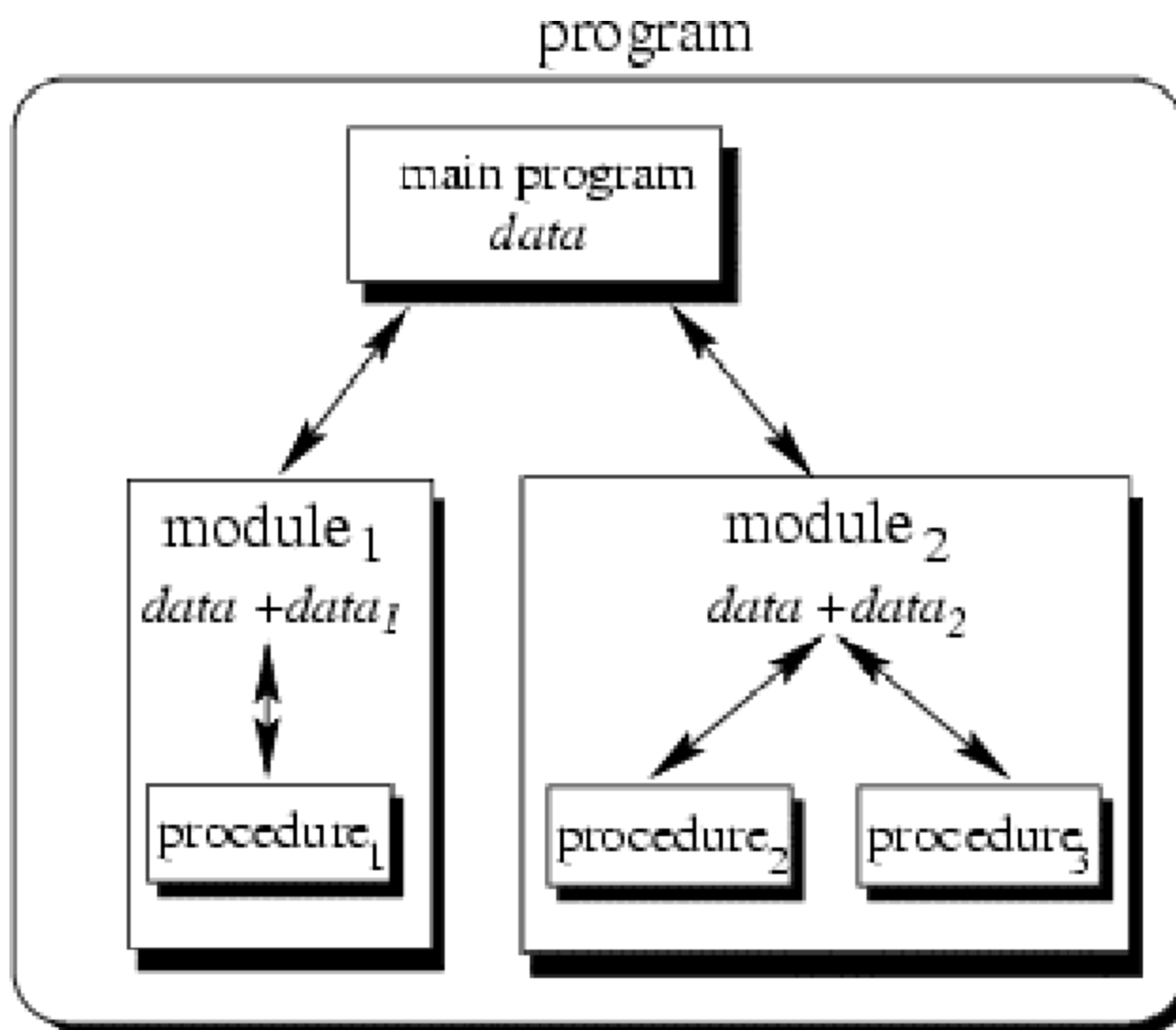
`#include MYHEADER`

`#include "MYHEADER"`

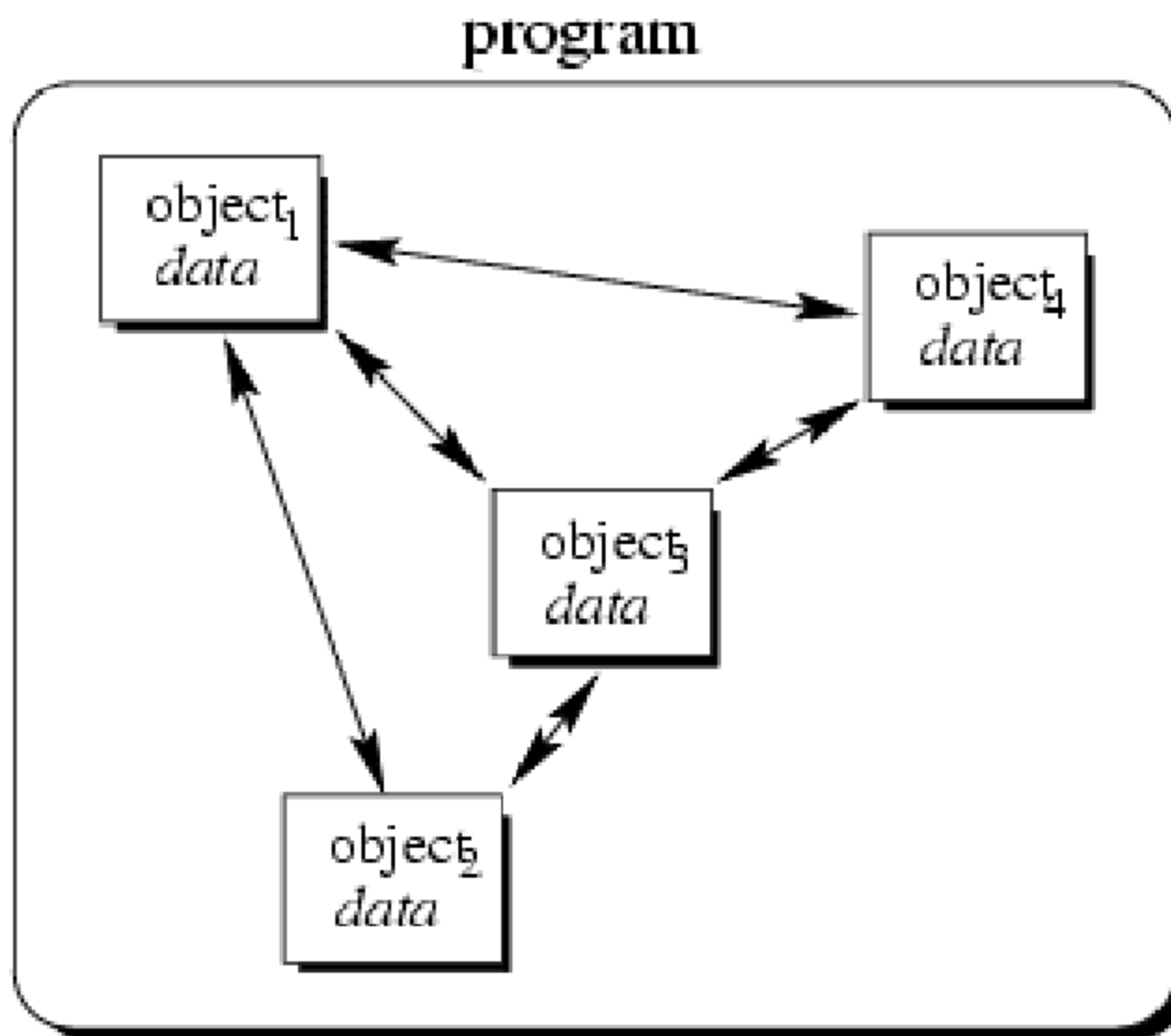
Procedurālā programmēšana



Modulārā programmēšana



Objektorientētā programmēšana



Objektorientētās programmēšanas pamatjēdzieni

n Objekts

n Ziņojums

n Klase

n Instance, eksemplārs

n Metode

n Abstrakcija

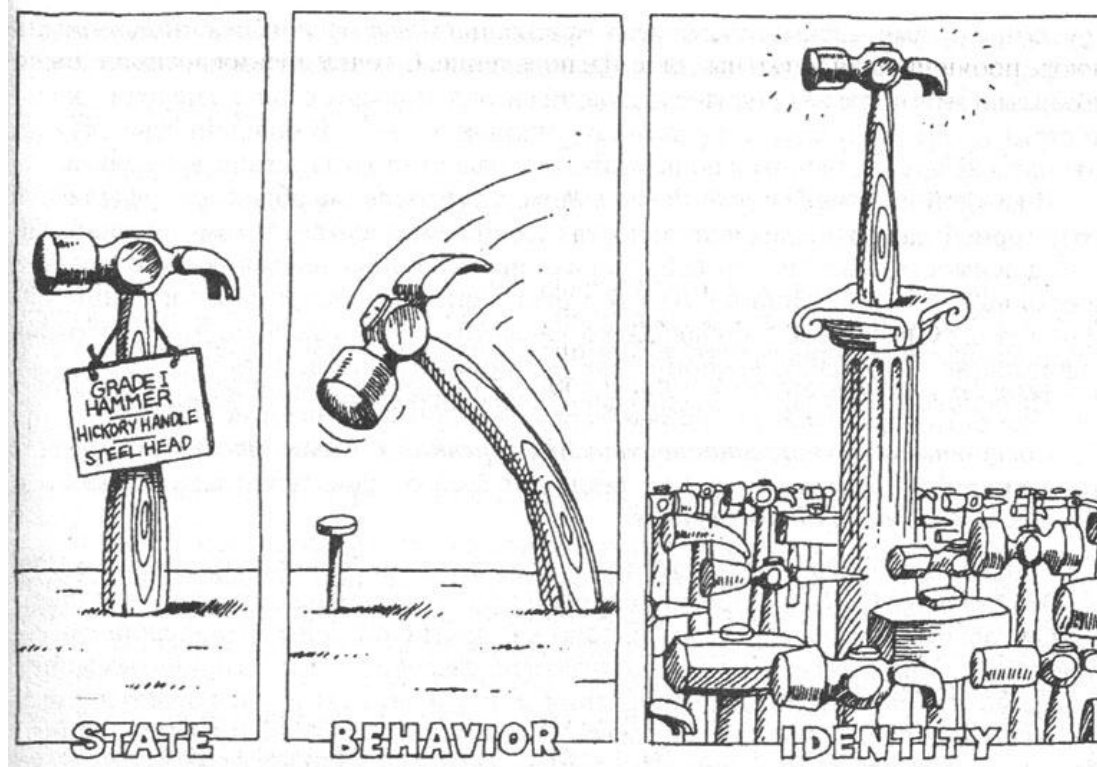
n Iekapsulēšana

n Mantošana

n Polimorfisms

Objekts

- Objekts ir klases eksemplārs
- Objektam ir identitāte, stāvoklis un uzvedība



Triangle t;

t.a = 50; t.b = 60; t.c = 35;

x = t.area();

// Objekts t

// Malu garumi-stāvoklis

// Laukums - uzvedība³³

Ziņojums

n Veids, kā objekti savā starpā sazinās – objekta metodes izsaukums

...

Word W;

W. OpenDocument ("c: \Docs\Letter. doc");

W. PrintDocument ();

...

Triangle t13;

...

s = t13. area();

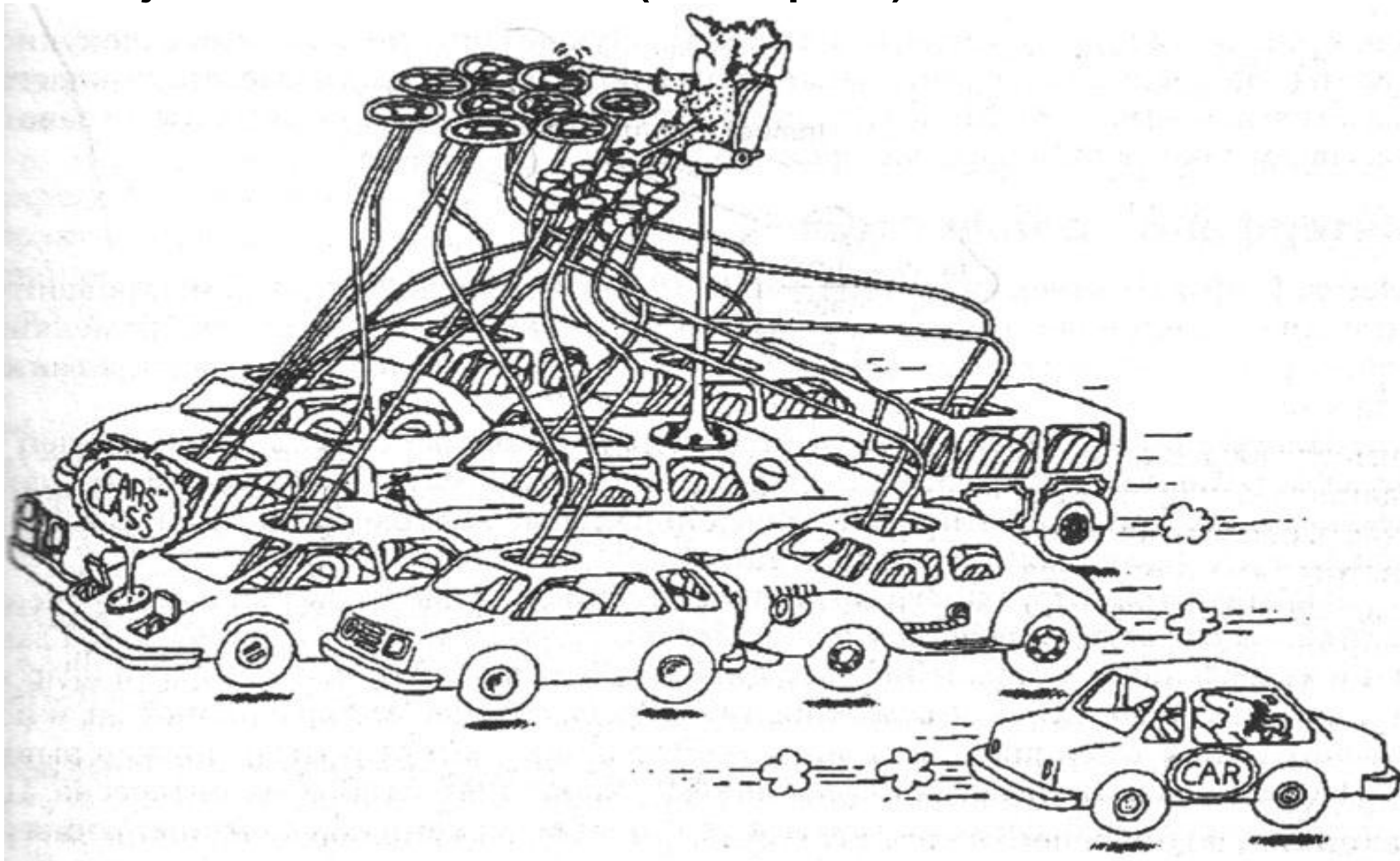
Metode

- n Definē algoritmu, kuru izpilda objekts, reaģējot uz ziņojumu
- n Metode ir kods, kas ir piesaistīts vai nu klasei (klases metode jeb statiskā metode), vai objektam (instances metode)

Klase

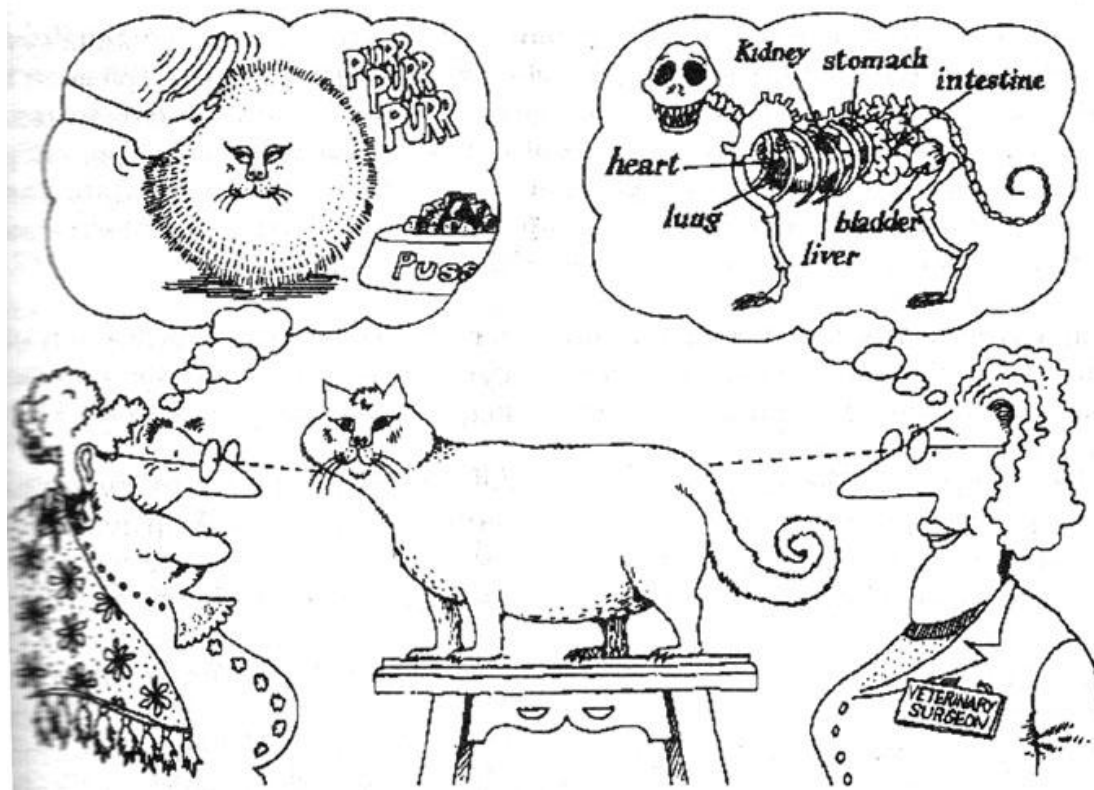
n Klase reprezentē objektu kopu ar līdzīgu struktūru un uzvedību

n Objekts ir klases instance (eksemplārs)



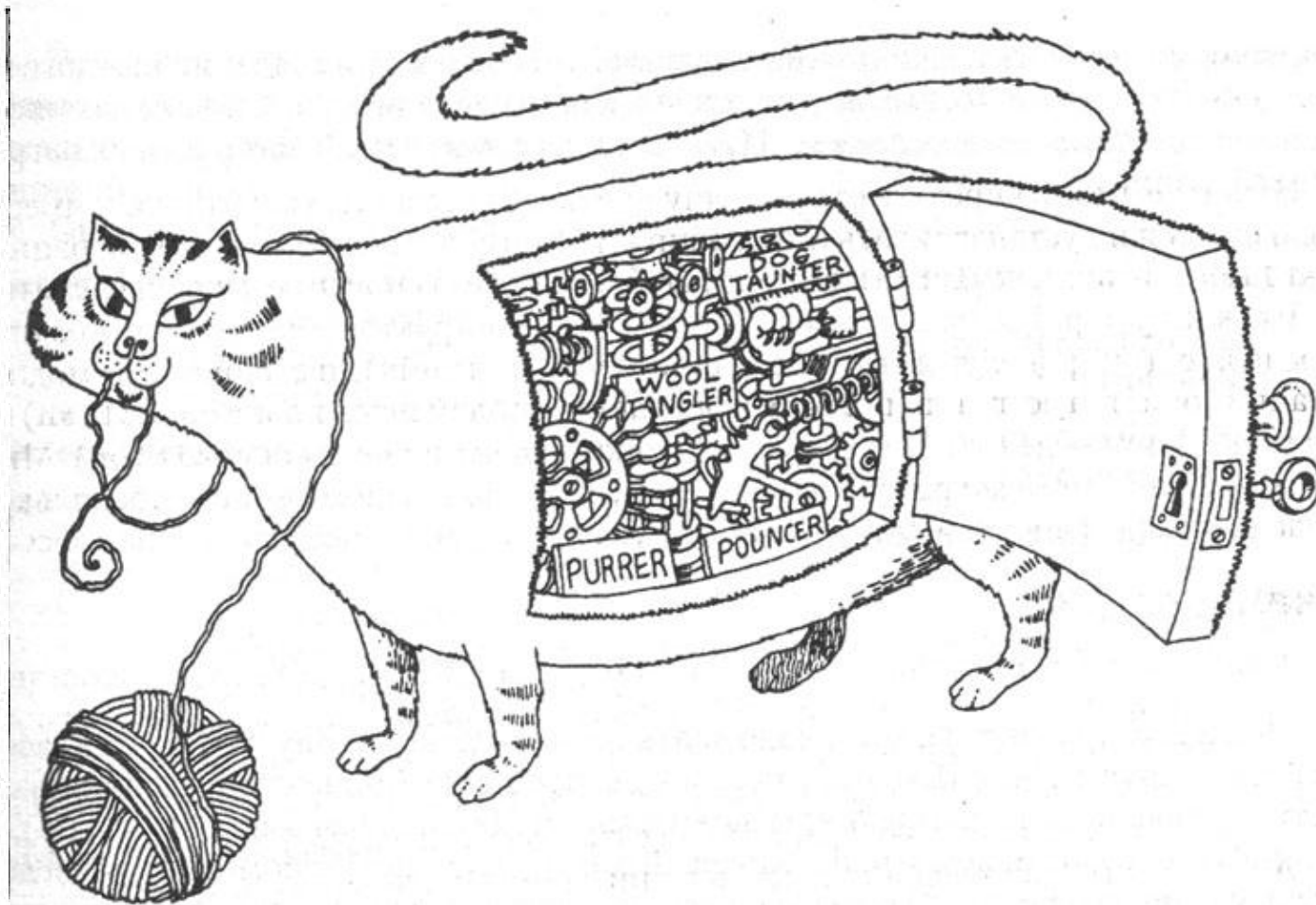
Abstrakcija

- n Vienkāršots apraksts vai skats uz kaut ko, kas uzsver tā būtiskākās raksturiezīmes vai mērķus, tajā pašā laikā atmetot detaļas, kas nav būtiskas vai ir traucējošas uztverei.
- n Izdala būtiskākos kāda objekta raksturojumus, kas to atšķir no visiem pārējiem objektiem, tādā veidā, no vērotāja viedokļa, precizē tā konceptuālās robežas.



Iekapsulēšana

n Iekapsulēšana apraksta objekta spēju paslēpt savus datus un metodes no ārienes



Iekapsulēšana valodā C++

Klases

```
class Triangle {  
public: int a, b, c;  
       float area();  
       int perimeter();  
};
```

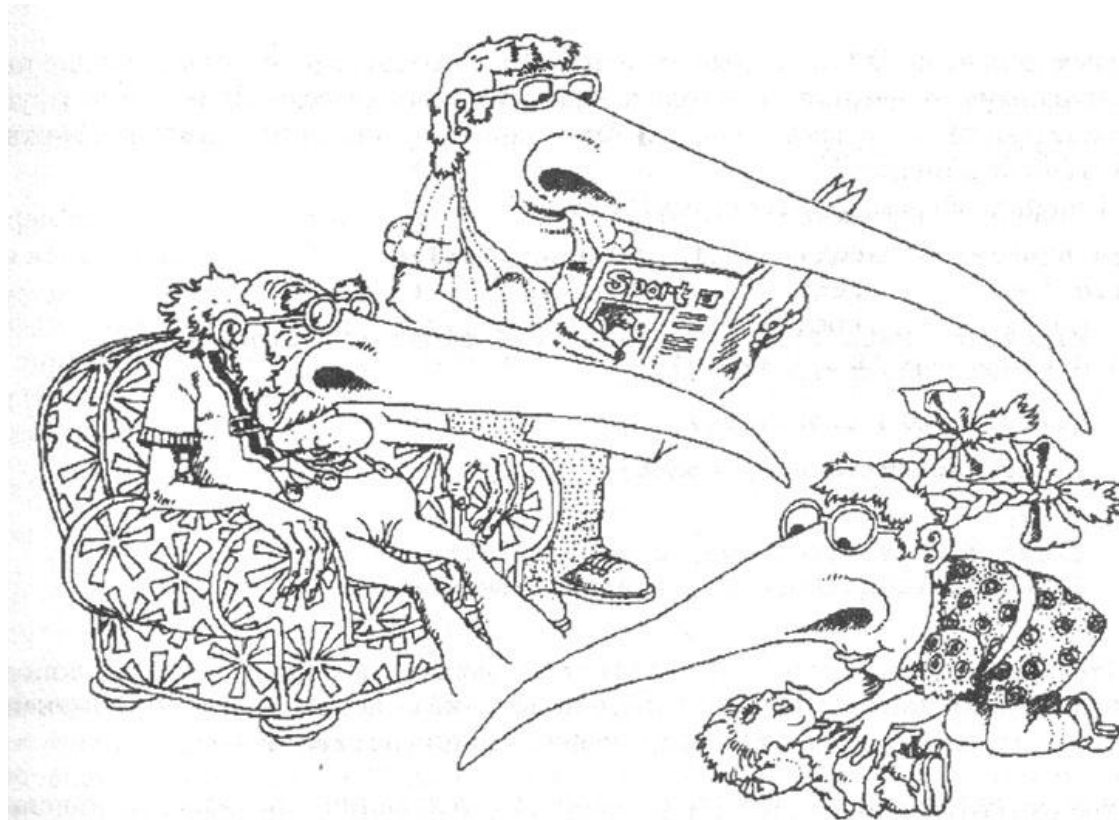
```
class Dog {  
private: char Name[12];  
public:  char* getName();  
};
```

Struktūras

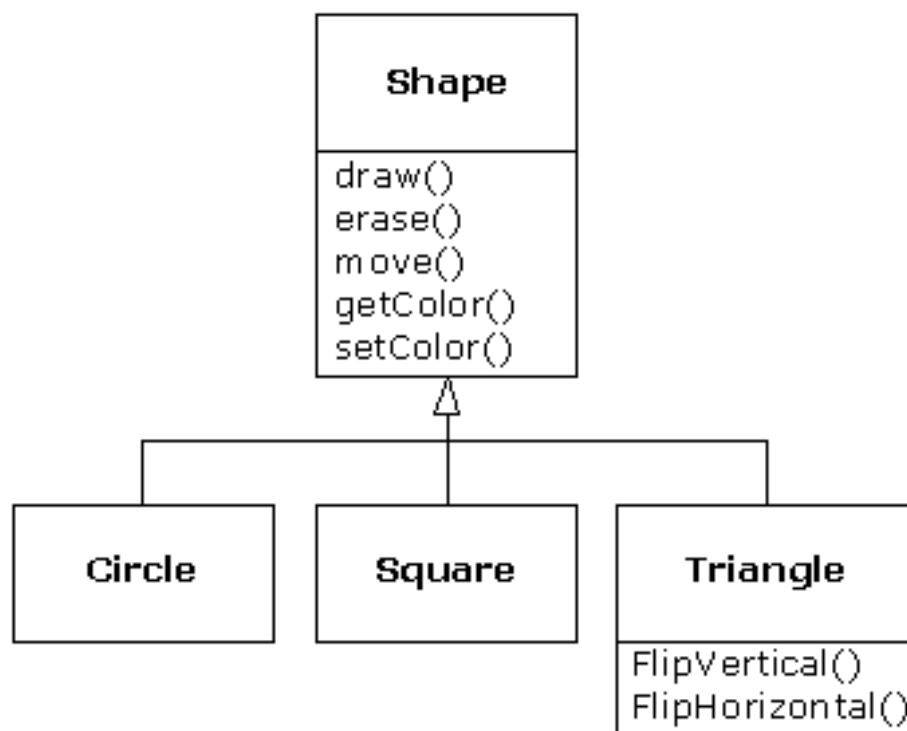
```
struct Record {  
    int    number;  
    char*  name;  
};
```


Mantošana

- n Apakšklase manto virsklases struktūru un uzvedību
- n Veids, kā formēt jaunas klases, izmanto iepriekš definētās klases, kur jaunās klases pārņem iepriekšējo klašu īpašības
- n Izmanto, lai veicinātu esošā koda izmantošanu ar nelielām vai vispār bez modifikācijām



Mantošana valodā C++



Polimorfisms

- n Vārdam 'polimorfisms' ir grieķu izcelsme un tā nozīme ir 'tāds, kuram ir vairākas formas'.
- n Objektorientētajā programmēšanā polimorfisma princips ir 'viens interfeiss, vairākas metodes'.
- n Valodā C++ polimorfismu visbiežāk lieto attiecībā uz funkcijām un operācijām, kad vienādi nosauktas funkcijas vai operācijas realizē dažādas darbības.

Operācija <<

```
cout << "Ievadiet paroli: ";  
...  
int n, flag;  
flag = n << 2;
```

Polimorfisms

```
class Square {  
public:  
    int side;  
    int area();  
    int perimeter();  
};
```

```
int Square::area()  
{  
    return side * side;  
}
```

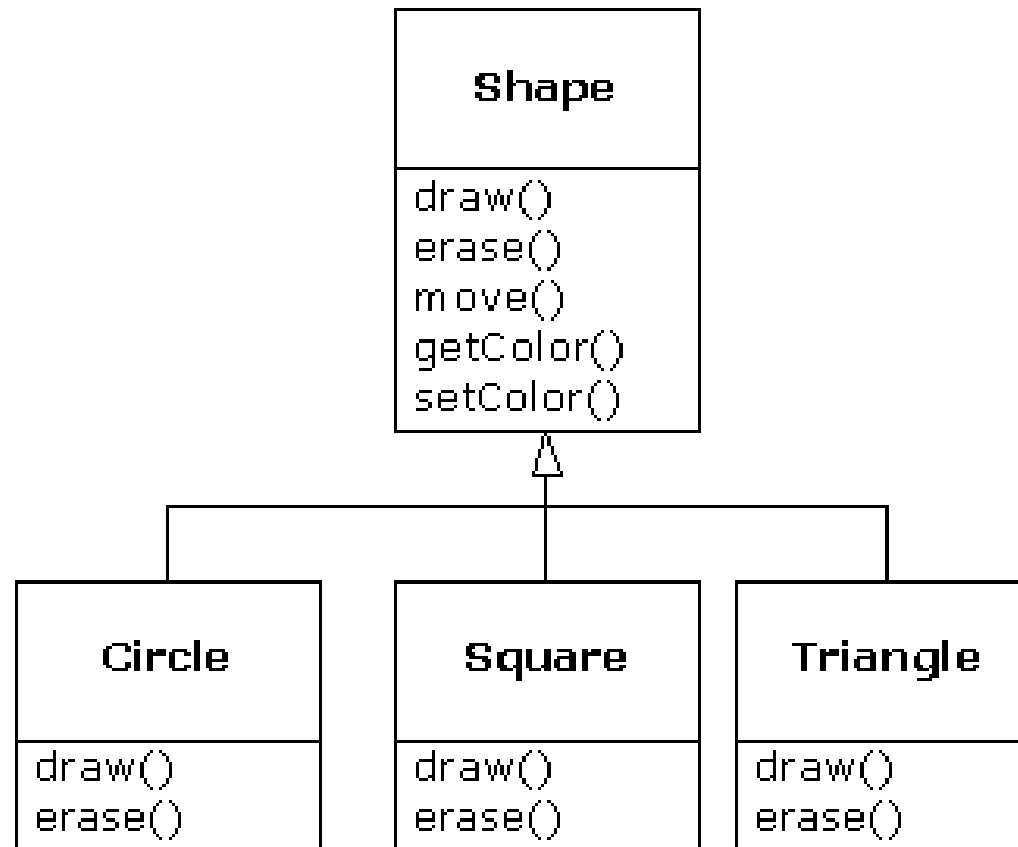
```
int Square::perimeter()  
{  
    return 4*side;  
}
```

```
class Circle {  
public:  
    int radius;  
    float area();  
    float perimeter();  
};
```

```
float Circle::area()  
{  
    return 3.14159*radius*radius;  
}
```

```
float Circle::perimeter()  
{  
    return (2 * 3.14159 * radius);  
}
```

Polimorfisms



Uzdevuma risināšanas soļi

- 1. Analizēt uzdevumu un identificēt galvenos objektus.**
- 2. Aprakstīt šo objektu iekšējo datu struktūru.**
- 3. Grupēt objektus, veidojot augstākas klases. Uzzīmēt klašu hierarhiju.**
- 4. Katrai klasei noteikt metodes un to pieejamības noteikumus.**
- 5. Izstrādāt kopējo uzdevuma risināšanas shēmu.**
- 6. Uzrakstīt katras metodes algoritmu.**
- 7. Pārskatīt un uzlabot klases un uzdevuma risināšanas shēmu.**