Prog1, C++ tárgyalás

Magasszintű programozási nyelvek 1

BSc előadás

Dr. Bátfai Norbert
egyetemi adjunktus
http://www.inf.unideb.hu/~nbatfai/
Debreceni Egyetem, Informatikai Kar,
Információ Technológia Tanszék
batfai.norbert@inf.unideb.hu
Skype: batfai.norbert

Prog1_6.ppt, v.: 0.0.9, 2012. 04. 10.
http://www.inf.unideb.hu/~nbatfai/#p1

Az óra blogja: http://progpater.blog.hu/
A Nokia Ovi store-ban is elérhető: http://store.ovi.com/content/100794

Felhasználási engedély

Bátfai Norbert
Debreceni Egyetem, Informatikai Kar, Információ Technológia Tanszék
<nbatfai@inf.unideb.hu, nbatfai gmail com>

Célok és tartalom

Előadás

a) Osztályok, objektumok
b) Másoló konstruktor, - értékadás, baráti, beágyazott osztályok
c) Öröklődés, a Liskov-féle helyettesítési elv
d) Virtuális függvények
e) C és C++ összehasonlítás

Labor

a) Qt-s példák
b) Visszatekintés: hálózati állatorvosi és a deriváló ló, a BB(2) megoldása

Laborkártyák

a) Példás kártyák

Otthoni opcionális feladat

a) A japán világbajnok HELIOS csapat szoftvereinek otthoni tanulmányozása.

Kapcsoldó videók, videómagyarázatok és blogok

http://progpater.blog.hu/2011/02/26/tan_csodallak_amde_nem_ertelek_de_kepzetem_hegyvolgyedet_bejarja
http://progpater.blog.hu/2011/03/03/fegyvert_a_nepnek
http://progpater.blog.hu/2011/03/05/figyelem_ez_nem_gyakorlat
http://progpater.blog.hu/2011/03/05/szonyegen_a_human_genom
http://progpater.blog.hu/2011/03/06/utam_a_csucsra
http://progpater.blog.hu/2011/03/31/imadni_fogjatok_a_c_t_egy_emberkent_tiszta_szivbol
http://progpater.blog.hu/2011/04/01/imadni_fogjak_a_c_t_egy_emberkent_tiszta_szivbol_2
http://progpater.blog.hu/2011/04/03/elmondtam_milliomezerszer_2
http://progpater.blog.hu/2011/04/03/a_nyolcadik_kilencedik_labor

Az írásbeli és a szóbeli vizsgán bármi (jegyzet, könyv, forráskód, számítógép mobiltelefon stb.) használható! (Az írásbeli vizsgán beszélni viszont tilos.) Hiszen az én feladatom az lesz, hogy eldöntsem, jól felkészült programozóval, vagy mennyire felkészült programozóval állok szemben.

Minimális gyakorlati cél

A hallgató tudja módosítani a bevezető Qt példákat:

svn co svn://hallg.inf.unideb.hu:2005/bevezetes

(az elsoQt/{Frak|Sejtauto|GenAblak|masodikQt/{Frak…}…})

Illetve az LZW fás példákat (legalább z1.cpp, z3.cpp, z6.cpp)

(elsoC++/ziv)

Az svn tároló elérése kapcsán:

http://progpater.blog.hu/2011/02/05/az_elso_labor/fullcommentlist/1#c12856691

Minimális elméleti cél

1) Másoló konstruktor, másoló értékadás
2) Függvény, operátor túlterhelés
3) Öröklődés, a Liskov-féle helyettesítési elv
4) Virtuális függvények, polimorfizmus
5) C és C++ összehasonlítása
6) Dinamikus tárkezelés áttekintése: a malloc()-tól a new-ig (a többdimenziós példákon át is), példát mutatva a memória szivárgásra
7) A védendő C++ forrás ismerete: http://progpater.blog.hu/2012/04/10/imadni_fogjak_a_c_t_egy_emberkent_tiszta_szivbol_4

Visszatekintés: példányosítás

```
Osztaly peldany;                                    (példány a veremben)

peldany.metodus();
```

```
Osztaly *peldany;                                   (példány a halmon)
peldany = new Osztaly();

peldany->metodus();   // (*peldany).metodus();

delete peldany;
```

Másoló konstruktor

```
#include <iostream>

class Verem
{

public:
Verem (int m = 1024):meret (m), verem (new char[m])
  {
    sp = -1;
  }
   ~Verem ()
  {
    delete[]verem;
  }
  char pop ()
  {
    return verem[sp--];
  }
  void push (char c)
  {
    verem[++sp] = c;
  }
  int getMeret ()
  {
    return meret;
  }

private:
  char *verem;
  int meret;
  int sp;
};
```

```
int
main (int argc, char *argv[])
{

  Verem v (512);

  v.push ('0');
  v.push ('1');

  std::cout << v.pop () << std::endl;
  std::cout << v.pop () << std::endl;
  std::cout << v.getMeret () << std::endl;

  return 0;
}
```

```
nbatfai@hallg:~$ g++ verem.cpp -o verem
nbatfai@hallg:~$ ./verem
1
0
512
```

Másoló konstruktor

```
class Verem
{

public:
Verem (int m = 1024):meret (m), verem (new char[m])
  {
    sp = -1;
  }
   ~Verem ()
  {
    delete[]verem;
  }
  char pop ()
  {
    return verem[sp--];
  }
  void push (char c)
  {
    verem[++sp] = c;
  }
  int getMeret ()
  {
    return meret;
  }

private:
  char *verem;
  int meret;
  int sp;
};
```

```
int
main (int argc, char *argv[])
{

  Verem v (512);
  Verem u = v;        // Verem u(v);

  v.push ('0');
  v.push ('1');

  std::cout << v.pop () << std::endl;
  std::cout << v.pop () << std::endl;
  std::cout << v.getMeret () << std::endl;

  return 0;
}
```

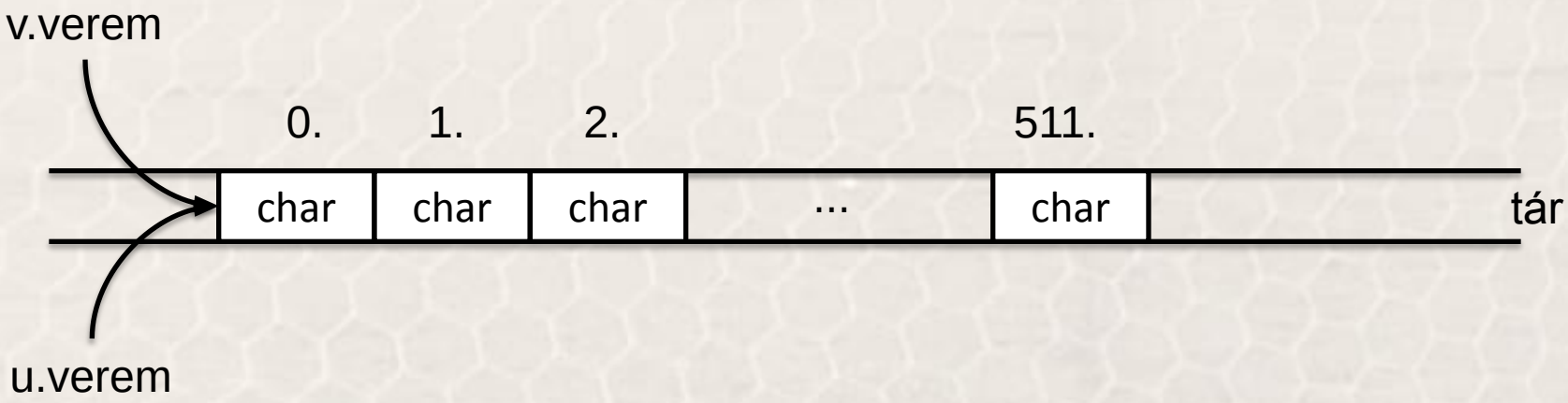
Másoló konstruktor

```
int
main (int argc, char *argv[])
{

  Verem v (512);
  Verem u = v;

  v.push ('0');
  v.push ('1');

  std::cout <<
  std::cout <<
  std::cout <<

  return 0;
}
```

```
[morpheus@zion morpheus]$  g++ verem1.cpp -o verem
[morpheus@zion morpheus]$ ./verem
1
0
512
Szegmens hiba

nbatfai@hallg:~$ g++ verem1.cpp -o verem
nbatfai@hallg:~$ ./verem
1
0
512
*** glibc detected *** ./verem: double free or corruption (top):
0x0000000000602010 ***
======= Backtrace: =========
/lib/libc.so.6[0x7f20feb20928]
```

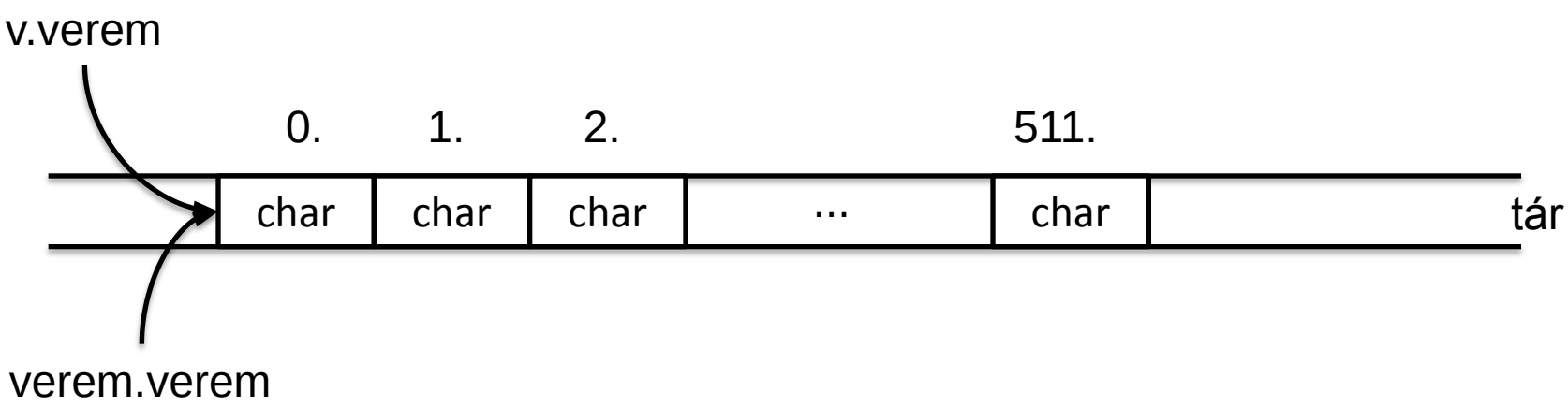
Másoló konstruktor

```
int
main (int argc, char *argv[])
{

  Verem v (512);
  Verem u = v;

  v.push ('0');
  v.push ('1');

  std::cout << v.pop () << std::endl;
  std::cout << v.pop () << std::endl;
  std::cout << v.getMeret () << std::endl;

  return 0;
}
```



Sekély másolás

```
void
ertekSzerint (Verem verem)
{
  verem.push ('0');
  verem.push ('1');

  std::cout << verem.pop () << std::endl;
  std::cout << verem.pop () << std::endl;
  std::cout << verem.getMeret () << std::endl;
}

int
main (int argc, char *argv[])
{
  V

  e

  V
  V

  s
  s
  s

  r
}
```



Mély másolás

```
class Verem
{
 public:
  Verem (int m = 1024):meret (m), verem (new char[m])
  {
    sp = -1;
  }
  Verem (Verem& v):meret (v.meret), verem (new char[v.meret])
  {
    sp = v.sp;

    for (int i = 0; i < v.meret; ++i)
      verem[i] = v.verem[i];
  }
  ~Verem ()
  {
    delete[]verem;
  }
  char pop ()
  {
    return verem[sp--];
  }
  void push (char c)
  {
    verem[++sp] = c;
  }
  int getMeret ()
  {
    return meret;
  }
  int getDarab ()
  {
    return sp + 1;
  }

 private:
  char *verem;
  int meret;
  int sp;
};
```

```
void
ertekSzerint (Verem verem)
{

  verem.push ('2');
  verem.push ('3');

  while(verem.getDarab ())
    std::cout << verem.pop () << std::endl;

}

int
main (int argc, char *argv[])
{

  Verem v (512);

  v.push ('0');
  v.push ('1');

  ertekSzerint (v);

  while(v.getDarab ())
    std::cout << v.pop () << std::endl;

  return 0;
}
```

Mély másolás

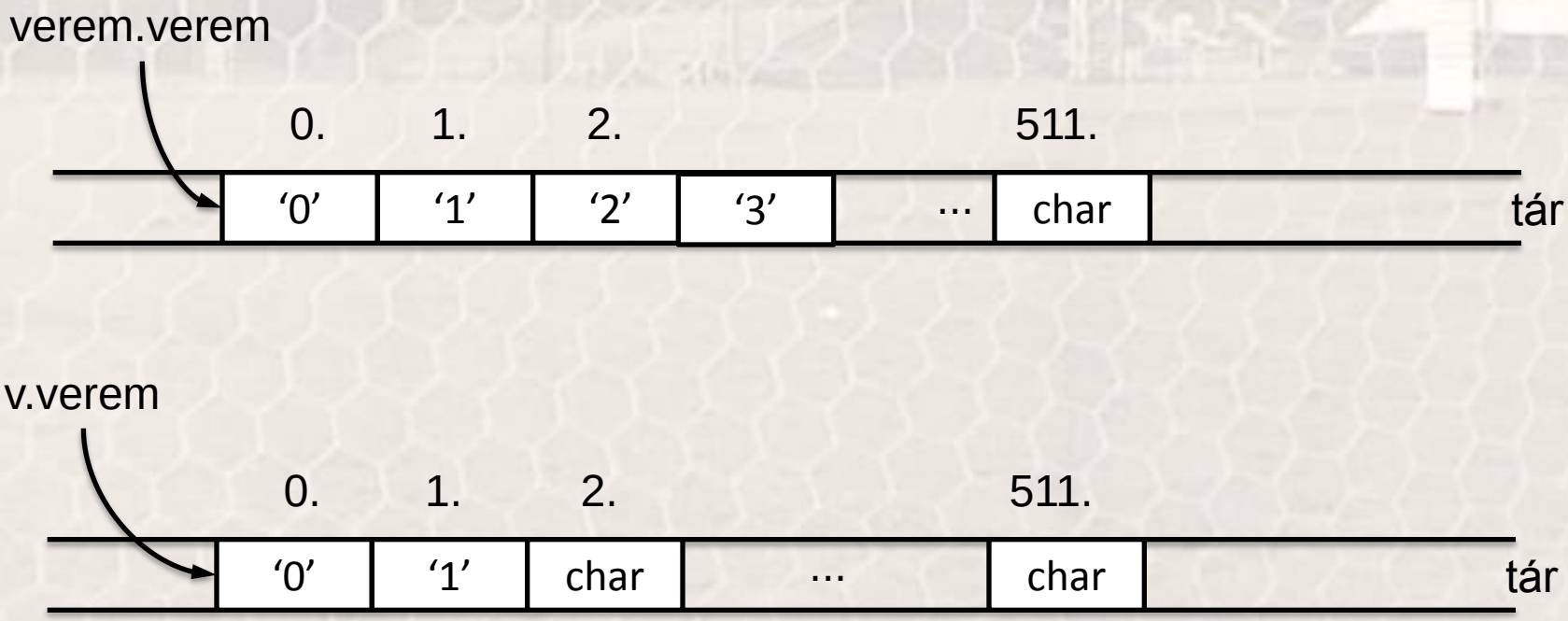
```
class Verem
{
 public:
  Verem (int m = 1024):meret (m), verem (new char[m])
  {
    sp = -1;
  }
  Verem (Verem& v):meret (v.meret), verem (new char[v.meret])
  {
    sp = v.sp;

    for (int i = 0; i < v.meret; ++i)
      verem[i] = v.verem[i];
  }
  ~Verem ()
  {
    delete[]verem;
  }
  char pop ()
  {
    return verem[sp--];
  }
  void push (char c)
  int meret;
  int sp;
};
```

```
nbatfai@hallg:~$ g++ verem4.cpp -o verem
nbatfai@hallg:~$ ./verem
3
2
1
0
1
0
```

```
void
ertekSzerint (Verem verem)
{

  verem.push ('2');
  verem.push ('3');

  while(verem.getDarab ())
    std::cout << verem.pop () <<

}

int
main (int argc, char *argv[])
{

  Verem v (512);

  v.push ('0');
  v.push ('1');

  ertekSzerint (v);

  while(v.getDarab ())
    std::cout << v.pop () << std:

  return 0;
}
```

Mély másolás

```
Verem (Verem& v):meret (v.meret), verem (new char[v.meret])
{
  sp = v.sp;

  for (int i = 0; i < v.meret; ++i)
    verem[i] = v.verem[i];
}
```



std::stack

```
#include <iostream>
#include <stack>

int
main (int argc, char *argv[])
{

  std::stack <char> v;

  v.push ('0');
  v.push ('1');

  std::cout << v.size () << std::endl;
  std::cout << v.top () << std::endl;
  v.pop();
  std::cout << v.top () << std::endl;
  v.pop();
  std::cout << v.size () << std::endl;

  return 0;
}
```

```
nbatfai@hallg:~$ g++ verem5.cpp -o verem
nbatfai@hallg:~$ ./verem
2
1
0
0
```

Mély másolás

```
#include <iostream>
#include <stack>

int
main (int argc, char *argv[])
{

  std::stack < char >v;

  v.push ('0');
  v.push ('1');

  std::stack < char >u = v;

  u.pop ();

  std::cout << v.size () << std::endl;
  std::cout << u.size () << std::endl;

  return 0;
}
```

```
nbatfai@hallg:~$ g++ verem6.cpp -o verem
nbatfai@hallg:~$ ./verem
2
1
```

Mély másolás

```
#include <iostream>
#include <stack>

void ertekSzerint (std::stack <char> verem)
{
  verem.push ('2');
  std::cout << verem.size () << std::endl;
}

int
main (int argc, char *argv[])
{

  std::stack < char >v;

  v.push ('0');
  v.push ('1');

  ertekSzerint (v);

  std::cout << v.size () << std::endl;

  return 0;
}
```

```
nbatfai@hallg:~$ g++ verem7.cpp -o verem
nbatfai@hallg:~$ ./verem
3
2
```

Ismétlés: laborkártya

Lefordul? Ha igen, mennyit ír ki?

```
#include <stdio.h>

int
main (void)
{
  int a = 5;
  int *ap = &a;
  int &ar = a;
  int *ap2 = &ar;

  ++a;

  ++*ap;

  ++ar;

  ++*ap2;

  printf ("a=%d\n", a);

  return 0;
}
```

```
[morpheus@zion morpheus]$ g++ r.c -o r
[morpheus@zion morpheus]$ ./r
a=9
```

```
int a = 5;
int &ar = a;
int &arr = ar;
++arr;
```

sekély másolás

a=?

Sekély másolás

```
int a = 5;
int &ar = a;
int &arr = ar;
++arr;

a=?
```

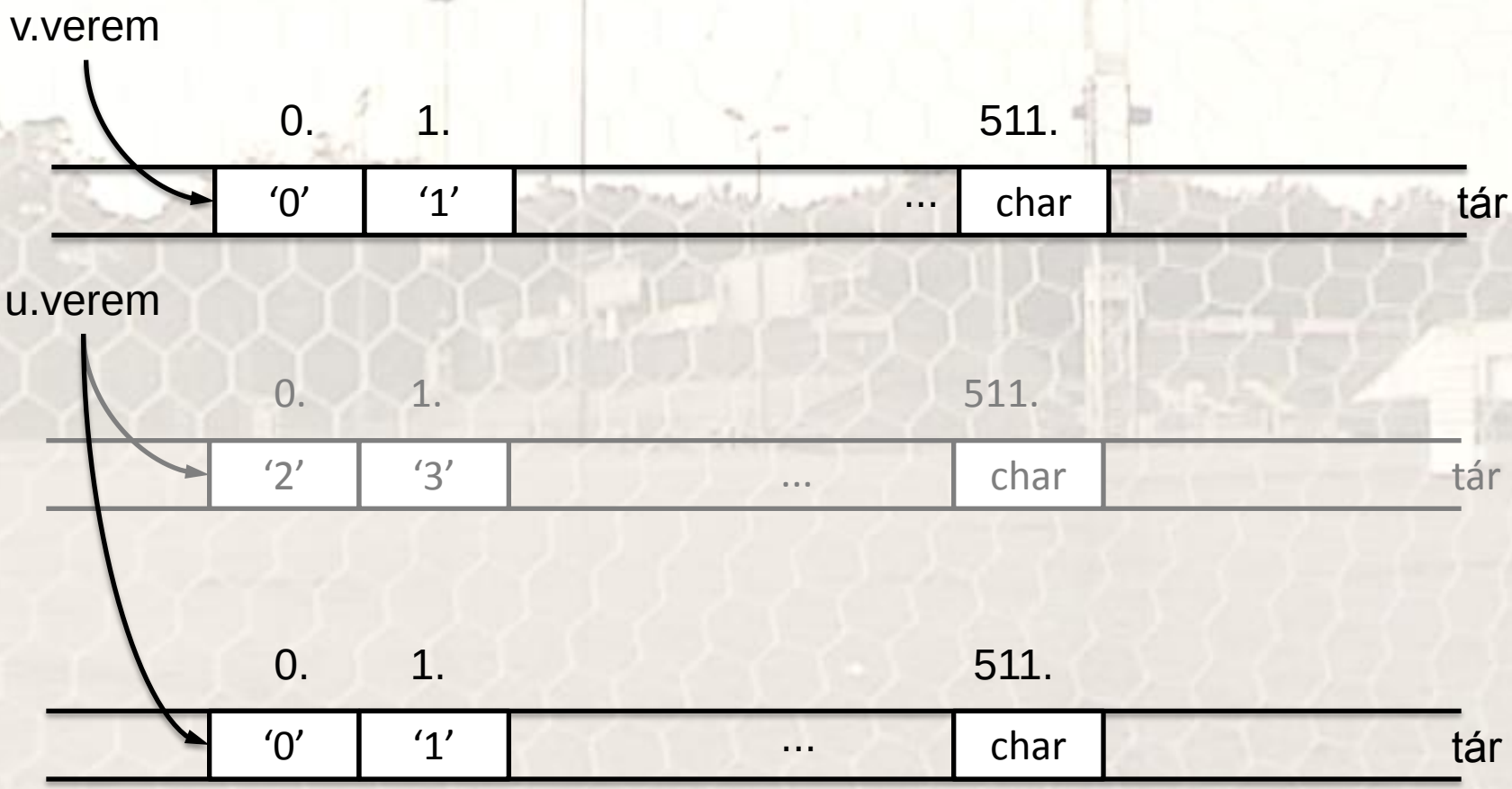
Másoló értékadás

```
Verem& operator=(Verem& v)
{
  char *ujverem = new char[v.meret];
  sp = v.sp;
  meret = v.meret;

  for (int i = 0; i < v.meret; ++i)
    ujverem[i] = v.verem[i];

  delete [] verem;
  verem = ujverem;

  return *this;
}
```

```
main (int argc, char *argv[])
{

  Verem v (512);

  v.push ('0');
  v.push ('1');

  Verem u;

  u.push ('2');
  u.push ('3');

  u = v;

  while(u.getDarab ())
    std::cout << "u: " << u.pop () << std::endl;

  while(v.getDarab ())
    std::cout << "v: " <<  v.pop () << std::endl;
```

```
nbatfai@hallg:~$ g++ verem9.c
nbatfai@hallg:~$ ./verem
u: 1
u: 0
v: 1
v: 0
```

Másoló értékadás



A másoló értékadás (és másoló konstruktor) letiltása

```
class SoccerAgent {
public:
    friend class BasicClient;

protected:
    //! interface to the rcssserver
    BasicClient * M_client;

private:

    // nocopyable
    SoccerAgent( const SoccerAgent & );
    SoccerAgent & operator=( const SoccerAgent & );

public:

    /*!
      \brief nothing to do. just set NULL to M_client
     */
    SoccerAgent();
```

this, *this

```
SejtSzal::SejtSzal(bool ***racsok, int szelesseg, int magassag, int varakozas, Sej
{
```

```
void FrakAblak::mouseReleaseEvent(QMouseEvent* event) {

    if(szamitasFut)
        return;

    szamitasFut = true;

    double dx = (b-a)/szelesseg;
    double dy = (d-c)/magassag;

    double a = this->a+x*dx;
    double b = this->a+x*dx+mx*dx;
    double c = this->d-y*dy-my*dy;
    double d = this->d-y*dy;

    this->a = a;
    this->b = b;
    this->c = c;
    this->d = d;

    delete mandelbrot;
    mandelbrot = new FrakSzal(a, b, c, d, szelesseg, magassag, iteraciosHatar, this);
    mandelbrot->start();

    update();
}
```

this, *this

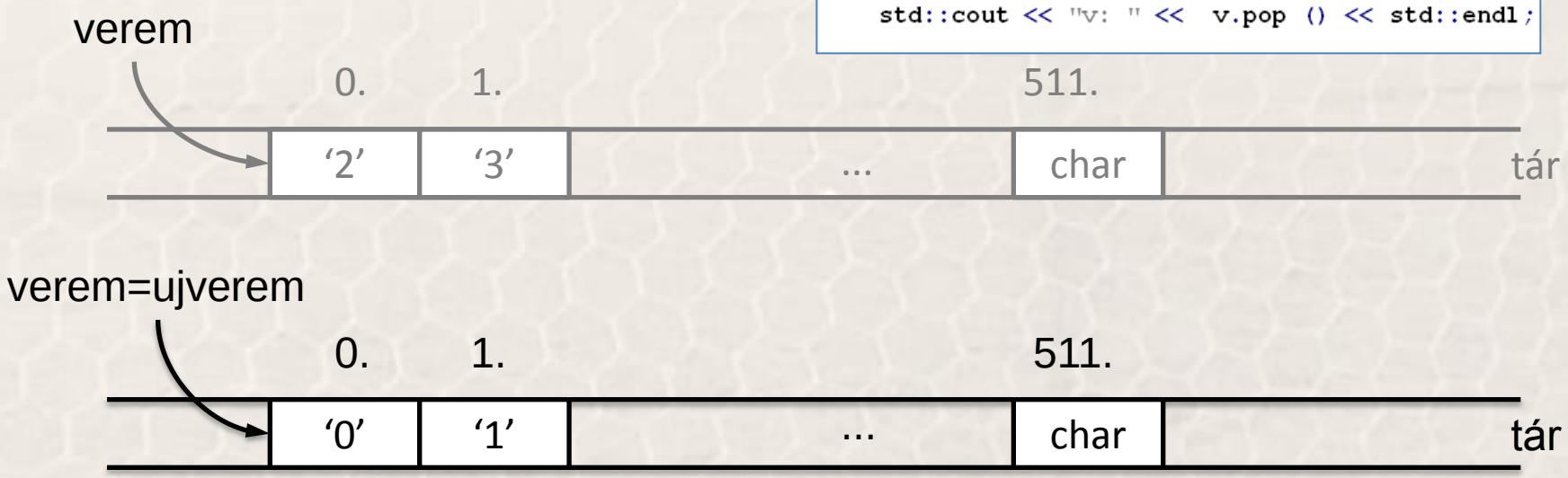
```
int
main ()
{
    BinFa binFa;
    std::string s;

    while (std::cin >> s)
    {
        binFa << s;
    }
```

```
class BinFa
{
public:
    BinFa ():csomopont(0), gyoker(0) {}

    void operator<<(std::string s)
    {
        int e;

        if (csomopont == NULL) {

            csomopont = new Csomopont (s);
            gyoker = csomopont;

        } else if ((e = csomopont->tartalma().compare(s)) == 0) {

            csomopont->novel();

        } else if (e > 0) {

            if (csomopont->balra() == NULL) {
                csomopont->balra(new Csomopont (s));
            } else {
                csomopont = csomopont->balra();
                *this << s;
            }
```

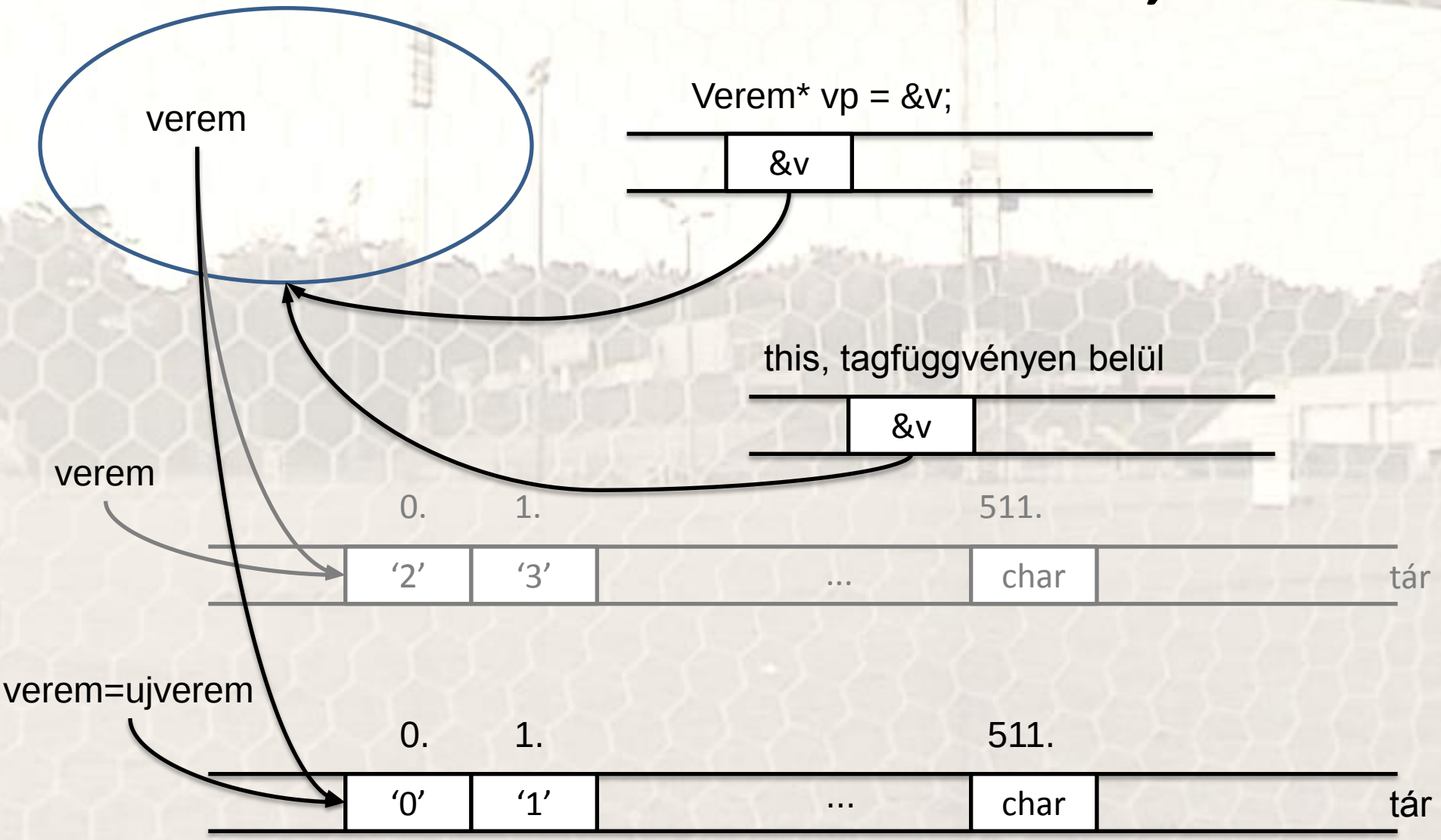
this, *this

```
Verem& operator=(Verem& v)
  {
    char *ujverem = new char[v.meret];
    sp = v.sp;
    meret = v.meret;

    for (int i = 0; i < v.meret; ++i)
      ujverem[i] = v.verem[i];

    delete [] verem;
    verem = ujverem;

    return *this;
  }
```

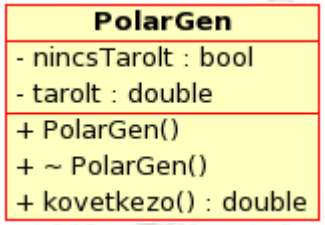
```
main (int argc, char *argv[])
{

  Verem v (512);

  v.push ('0');
  v.push ('1');

  Verem u;

  u.push ('2');
  u.push ('3');

  u = v;

  while(u.getDarab ())
    std::cout << "u: " << u.pop () << std::endl;

  while(v.getDarab ())
    std::cout << "v: " <<  v.pop () << std::endl;
```

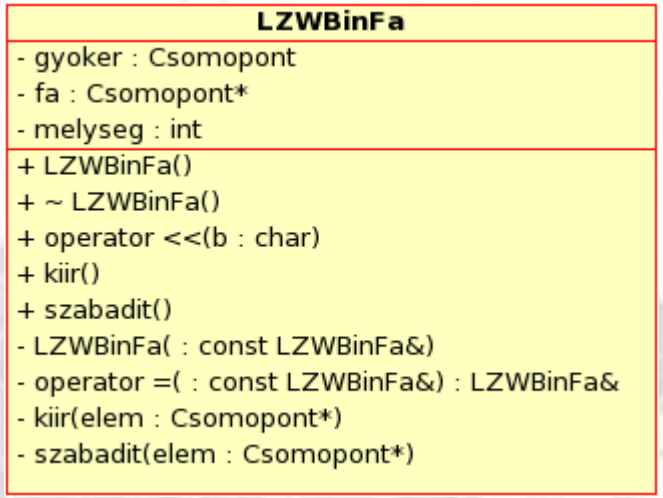


this, *this


verem
Verem* vp = &v;
&v
this, tagfüggvényen belül
&v
verem
0.
1.
511.
‘2’
‘3’
...
char
tár
verem=ujverem
0.
1.
511.
‘0’
‘1’
...
char
tár

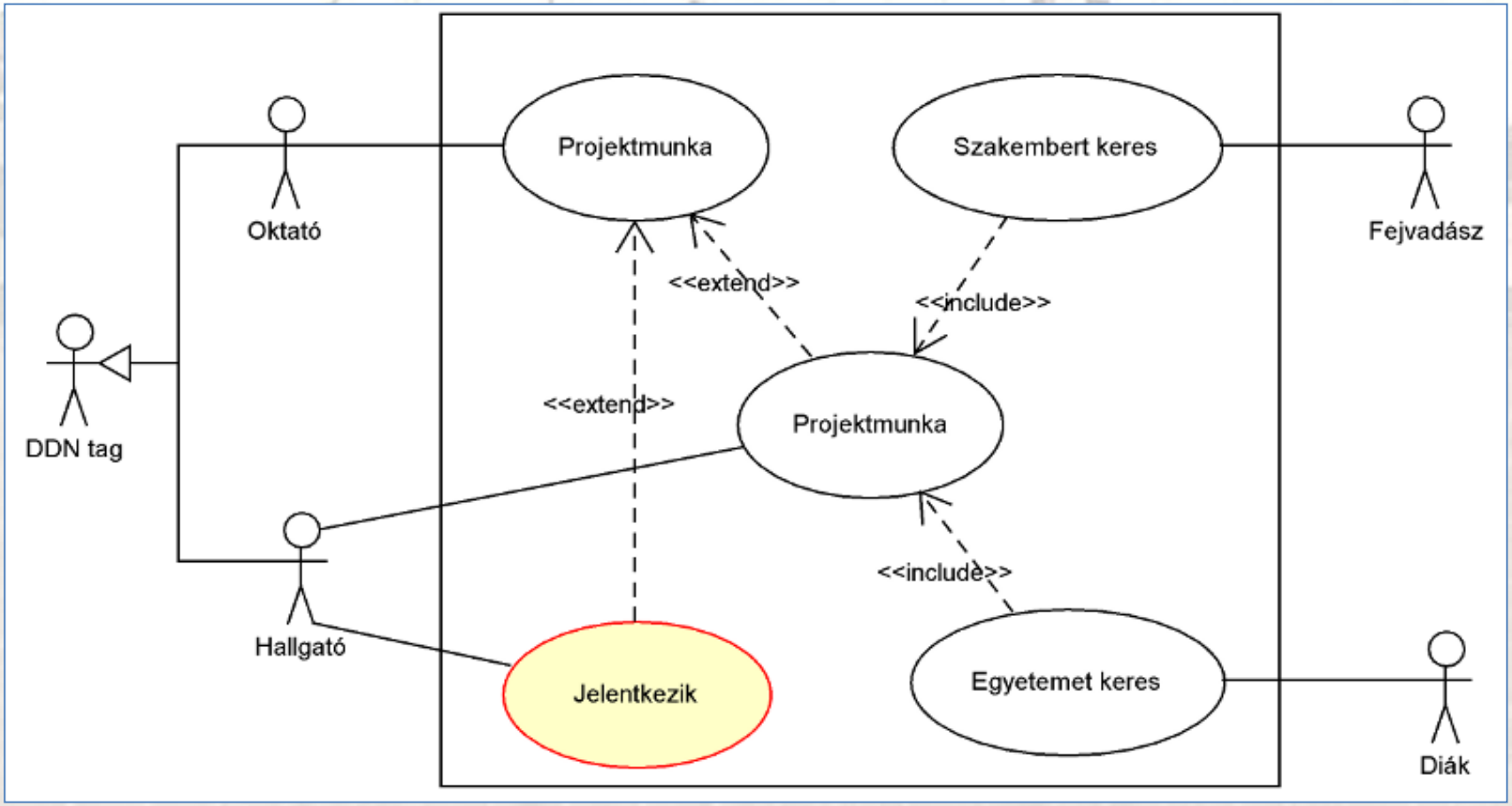
OO, UML


PolarGen
- nincsTarolt : bool
- tarolt : double
+ PolarGen()
+ ~ PolarGen()
+ kovetkezo() : double


LZWBinFa
- gyoker : Csomopont
- fa : Csomopont*
- melyseg : int
+ LZWBinFa()
+ ~ LZWBinFa()
+ operator <<(b : char)
+ kiir()
+ szabadit()
- LZWBinFa(: const LZWBinFa&)
- operator =(: const LZWBinFa&) : LZWBinFa&
- kiir(elem : Csomopont*)
- szabadit(elem : Csomopont*)

OMG UML: vizuális modellező nyelv

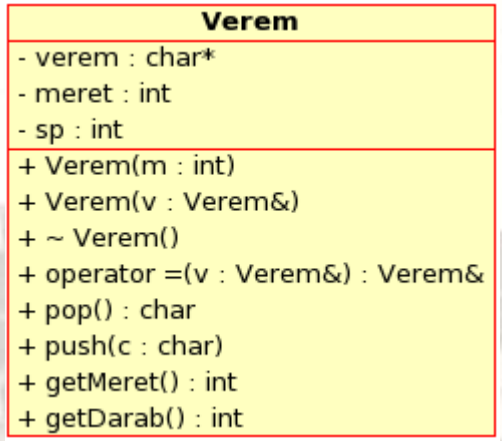
UML (OMG Unified Modeling Language) OO elvű modellezés
http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML
(Felépítés és működés modellezésére)



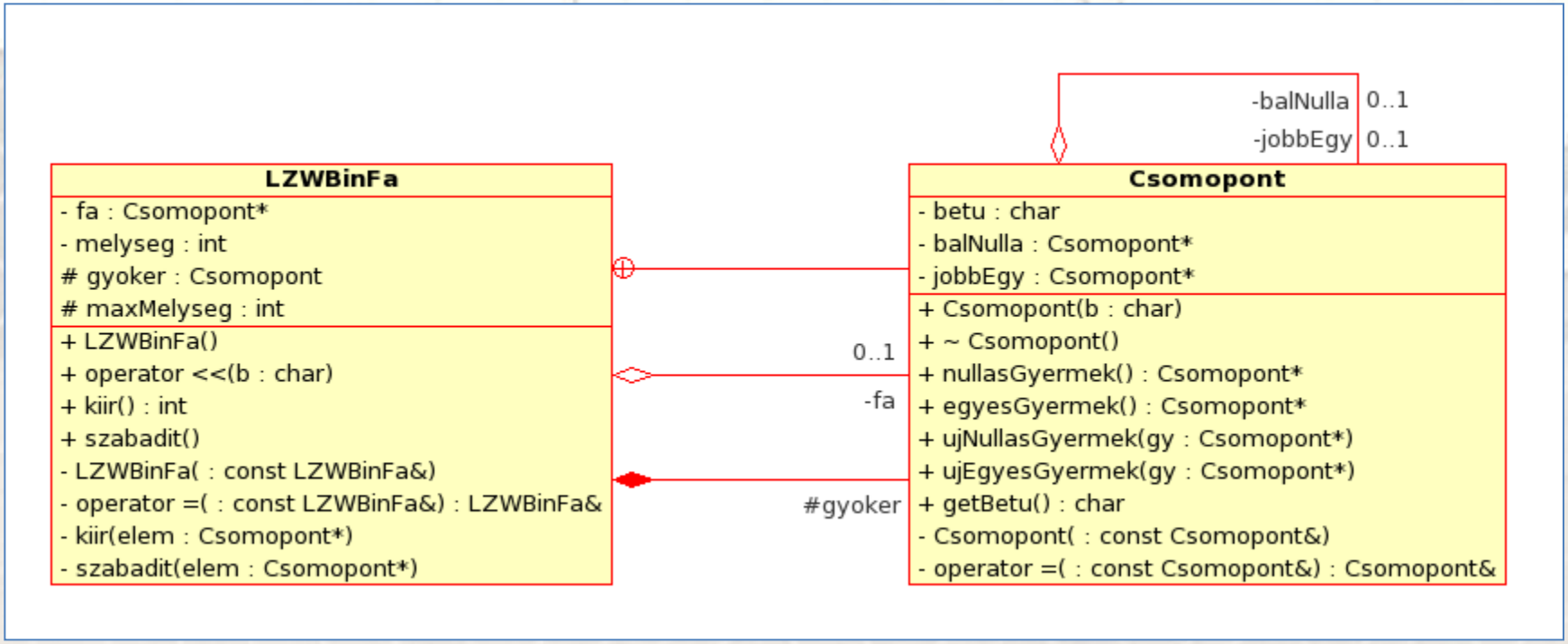
pl. használati esetek (működés)

Osztálydiagram

```
class Verem
{

public:
Verem (int m = 1024):meret (m), verem (new char[m])
  {
    sp = -1;
  }
  Verem (Verem& v):meret (v.meret), verem (new char[v.meret])
  {
    sp = v.sp;

    for (int i = 0; i < v.meret; ++i)
      verem[i] = v.verem[i];
  }
  ~Verem ()
  {
    delete[]verem;
  }
  Verem& operator=(Verem& v)
  {
    char *ujverem = new char[v.meret];
    sp = v.sp;
    meret = v.meret;

    for (int i = 0; i < v.meret; ++i)
      ujverem[i] = v.verem[i];

    delete [] verem;
    verem = ujverem;

    return *this;
  }
  char pop ()
  {
    return verem[sp--];
  }
  void push (char c)
  {
    verem[++sp] = c;
  }
  int getMeret ()
  {
    return meret;
  }
  int getDarab ()
  {
    return sp + 1;
  }

private:
  char *verem;
  int meret;
  int sp;
};
```

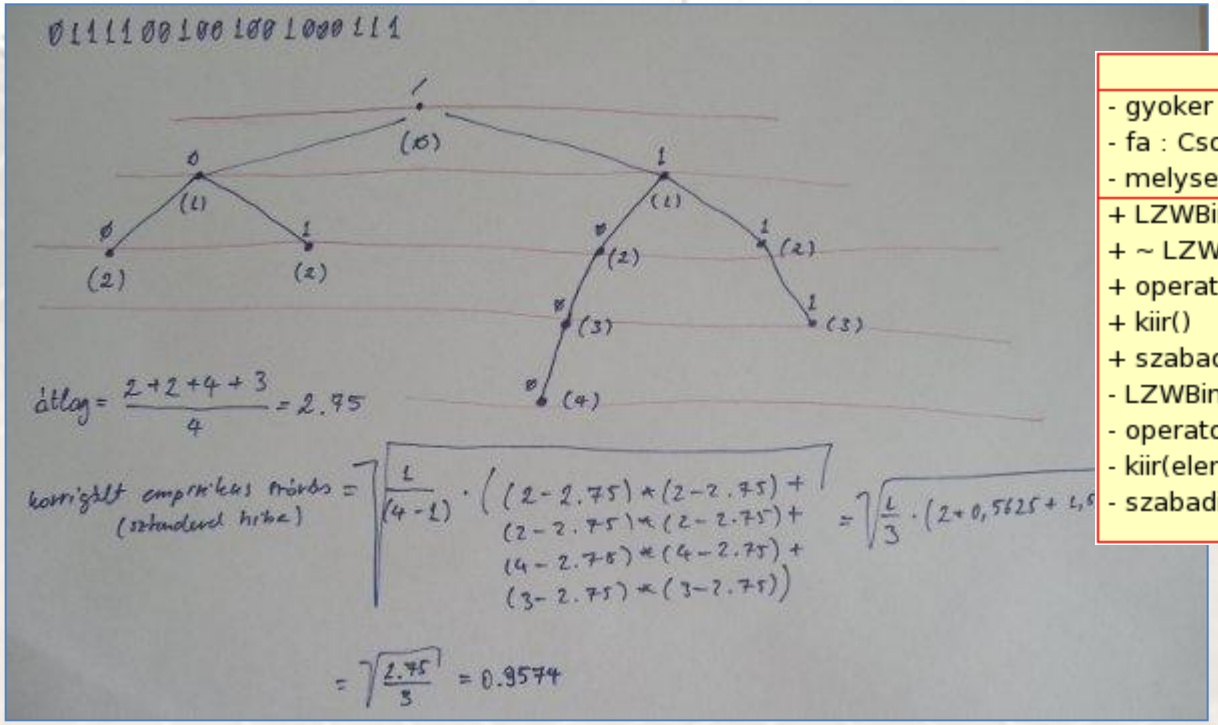


Asszociációk, aggregáció és kompozíció



Modellezés

„Valóság”



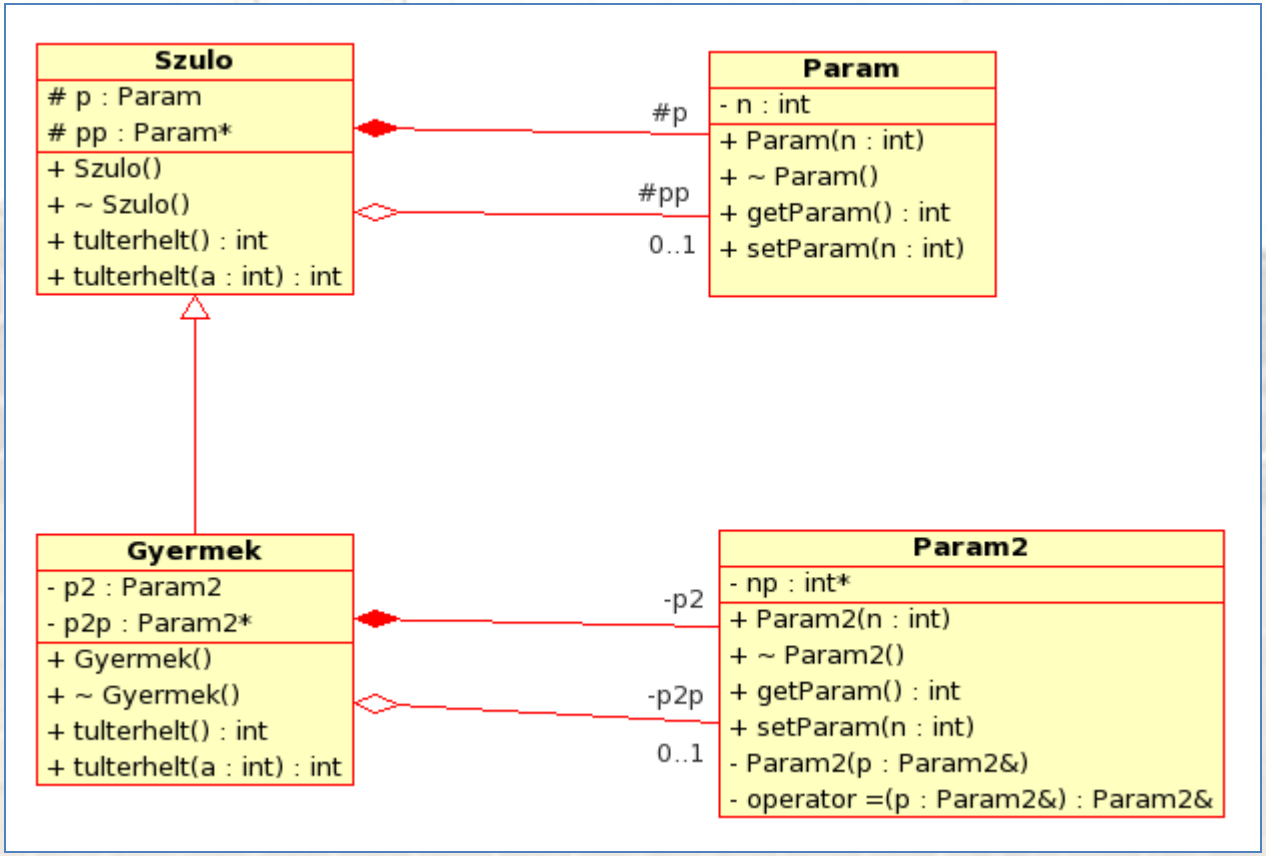
Modell

LZWBinFa
- gyoker : Csomopont - fa : Csomopont* - melyseg : int
+ LZWBinFa() + ~ LZWBinFa() + operator <<(b : char) + kiir() + szabadit() - LZWBinFa(: const LZWBinFa&) - operator =(: const LZWBinFa&) : LZWBinFa& - kiir(elem : Csomopont*) - szabadit(elem : Csomopont*)

```
#ifndef LZWBINFA_H
#define LZWBINFA_H

#include <iostream>

class LZWBinFa
{
public:
    LZWBinFa (): fa(&gyoker) {}
    ~LZWBinFa () {}
    void operator<<(char b);
    void kiir (void);
    void szabadit (void);

private:
    class Csomopont
    {
    public:
        Csomopont (char b = '/'):betu (b), balNulla (0), jobbEgy (0) {};
        ~Csomopont () {};
        Csomopont *nullasGyermek () const {return balNulla;}
        Csomopont *egyesGyermek ()  const {return jobbEgy;}
        void ujNullasGyermek (Csomopont * gy) {balNulla = gy;}
        void ujEgyesGyermek (Csomopont * gy) {jobbEgy = gy;}
    private:
        friend class LZWBinFa;
        char betu;
        Csomopont *balNulla;
        Csomopont *jobbEgy;
        Csomopont (const Csomopont &);
        Csomopont & operator=(const Csomopont &);
    };

    Csomopont gyoker;
    Csomopont *fa;
    int melyseg;

    LZWBinFa (const LZWBinFa &);
    LZWBinFa & operator=(const LZWBinFa &);

    void kiir (Csomopont* elem);
    void szabadit (Csomopont * elem);

};

#endif // LZWBINFA_H
```

```
void LZWBinFa::kiir (void)
{
    melyseg = 0;
    kiir (&gyoker);
}
void LZWBinFa::szabadit (void)
{
    szabadit (gyoker.jobbEgy);
    szabadit (gyoker.balNulla);
}

void LZWBinFa::kiir (Csomopont* elem)
{
    if (elem != NULL)
    {
        ++melyseg;
        kiir (elem->jobbEgy);
        // ez a postorder bejáráshoz képest
        // 1-el nagyobb mélység, ezért -1
        for (int i = 0; i < melyseg; ++i)
```

Öröklődés


Szulo
p : Param
pp : Param*
+ Szulo()
+ ~ Szulo()
+ tulterhelt() : int
+ tulterhelt(a : int) : int
Param
- n : int
+ Param(n : int)
+ ~ Param()
+ getParam() : int
+ setParam(n : int)
#p
#pp
0..1
Gyermek
- p2 : Param2
- p2p : Param2*
+ Gyermek()
+ ~ Gyermek()
+ tulterhelt() : int
+ tulterhelt(a : int) : int
Param2
- np : int*
+ Param2(n : int)
+ ~ Param2()
+ getParam() : int
+ setParam(n : int)
- Param2(p : Param2&)
- operator =(p : Param2&) : Param2&
-p2
-p2p
0..1

Öröklődés

```
class Param
{
public:
  Param (int n = 42):n (n){std::cout << "K:Param " << n << std::endl << std::flush;}
  ~Param (){std::cout << "D:Param" << std::endl << std::flush;}
  int getParam () const {return n;}
  void setParam (int n){this->n = n;}
private:
  int n;
};

class Param2
{
public:
  Param2 (int n = 42): np (new int(n)){std::cout << "K:Param2 " << n << std::endl << std::flush;}
  ~Param2 () {delete np; std::cout<<"D:Param2 " << std::endl << std::flush;}
  int getParam () const {return *np;}
  void setParam (int n){*this->np = n;}
private:
  int *np;
  Param2 (Param2& p);
  Param2& operator=(Param2& p);
};
```

Öröklődés

```
class Szulo
{
public:
  Szulo (): pp (&p){std::cout << "K:Szulo " << pp << std::endl << std::flush;}
  ~Szulo () {std::cout << "D:Szulo" << std::endl << std::flush;}
  virtual int tulterhelt () const {std::cout << "Szulo::tulterhelt" << std::endl << std::flush; return p.getParam();}
  int tulterhelt (int a) const {std::cout << "Szulo::tulterhelt2" << std::endl << std::flush; return a*p.getParam();}
protected:
  Param p;
  Param* pp;
};

class Gyermek : public Szulo
{
public:
  Gyermek (): Szulo (), p2p (&p2) {std::cout << "K:Gyermek " << pp << std::endl << std::flush;}
  ~Gyermek () {std::cout << "D:Gyermek" << std::endl << std::flush;}
  int tulterhelt () const {std::cout << "Gyermek::tulterhelt" << std::endl << std::flush; return p2.getParam();}
  int tulterhelt (int a) const {std::cout << "Gyermek::tulterhelt2" << std::endl << std::flush; return a*p2.getParam(
private:
  Param2 p2;
  Param2* p2p;
};
```

Liskov féle helyettesítési elv

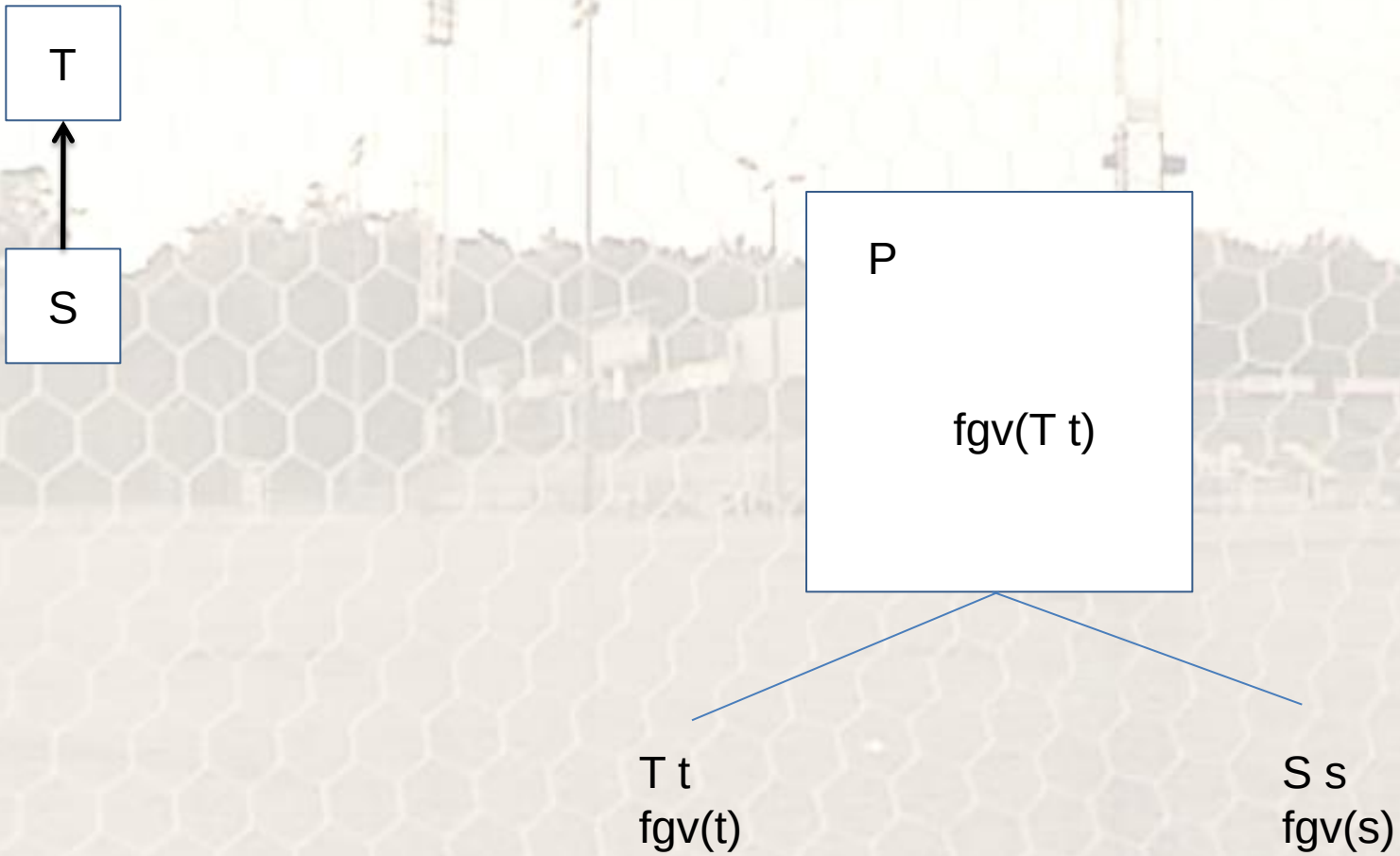
Barbara Liskov: Aata Abstraction and Hierarchy, OOPSLA '87 Addendum to the proceedings on Object-oriented programming systems, languages and applications (Addendum) ACM New York, NY, USA, 1987.
http://portal.acm.org/citation.cfm?doid=62138.62141

Liskov Substitution Principle (LSP)

3.3. Type Hierarchy

A type hierarchy is composed of subtypes and supertypes. The intuitive idea of a *subtype* is one whose objects provide all the behavior of objects of another type (the *supertype*) plus something extra. What is wanted here is something like the following substitution property [6]: If for each object o_1 of type S there is an object o_2 of type T such that for all programs P defined in terms of T, the behavior of P is unchanged when o_1 is substituted for o_2, then S is a subtype of T. (See also [2, 17] for other work in this area.)

Liskov féle helyettesítési elv



Polimorfizmus

```
void fgv(Szulo & szulo) {

  szulo.tulterhelt();

}

int
main (int argc, char *argv[])
{
  Szulo sz;
  sz.tulterhelt();
  fgv(sz);

  Gyermek gy;
  gy.tulterhelt();
  fgv(gy);

  Szulo * szp = new Gyermek();
  szp->tulterhelt();
  fgv(*szp);

  delete szp;

  return 0;
}
```

```
K:Param 42
K:Szulo 0x7fff1600ed30
Szulo::tulterhelt
Szulo::tulterhelt
K:Param 42
K:Szulo 0x7fff1600ed10
K:Param2 42
K:Gyermek 0x7fff1600ed10
Gyermek::tulterhelt
Szulo::tulterhelt
K:Param 42
K:Szulo 0x152d030
K:Param2 42
K:Gyermek 0x152d030
Szulo::tulterhelt
Szulo::tulterhelt
D:Szulo
D:Param
D:Gyermek
D:Param2
D:Szulo
D:Param
D:Szulo
D:Param
```

```
K:Param 42
K:Szulo 0x7fff40f38fb8
Szulo::tulterhelt
Szulo::tulterhelt
K:Param 42
K:Szulo 0x7fff40f38f88
K:Param2 42
K:Gyermek 0x7fff40f38f88
Gyermek::tulterhelt
Gyermek::tulterhelt
K:Param 42
K:Szulo 0x1c7b038
K:Param2 42
K:Gyermek 0x1c7b038
Gyermek::tulterhelt
Gyermek::tulterhelt
D:Szulo
D:Param
D:Gyermek
D:Param2
D:Szulo
D:Param
D:Szulo
D:Param
```

```
void fgv(Szulo & szulo) {

  szulo.tulterhelt();

}

int
main (int argc, char *argv[])
{
  Szulo sz;
  sz.tulterhelt();
  fgv(sz);

  Gyermek gy;
  gy.tulterhelt();
  fgv(gy);

  Szulo * szp = new Gyermek();
  szp->tulterhelt();
  fgv(*szp);

  delete szp;

  return 0;
}
```

Virtuális függvények

```
class Szulo
{
public:
  Szulo (): pp (&p){std::cout << "K:Szulo " << pp << std::endl << std::flush;}
  ~Szulo () {std::cout << "D:Szulo" << std::endl << std::flush;}
  virtual int tulterhelt () const {std::cout << "Szulo::tulterhelt" << std::endl << std::flush; return p.getParam();}
  int tulterhelt (int a) const {std::cout << "Szulo::tulterhelt2" << std::endl << std::flush; return a*p.getParam();}
protected:
  Param p;
  Param* pp;
};

class Gyermek : public Szulo
{
public:
  Gyermek (): Szulo (), p2p (&p2) {std::cout << "K:Gyermek " << pp << std::endl << std::flush;}
  ~Gyermek () {std::cout << "D:Gyermek" << std::endl << std::flush;}
  int tulterhelt () const {std::cout << "Gyermek::tulterhelt" << std::endl << std::flush; return p2.getParam();}
  int tulterhelt (int a) const {std::cout << "Gyermek::tulterhelt2" << std::endl << std::flush; return a*p2.getParam();
private:
  Param2 p2;
  Param2* p2p;
};
```

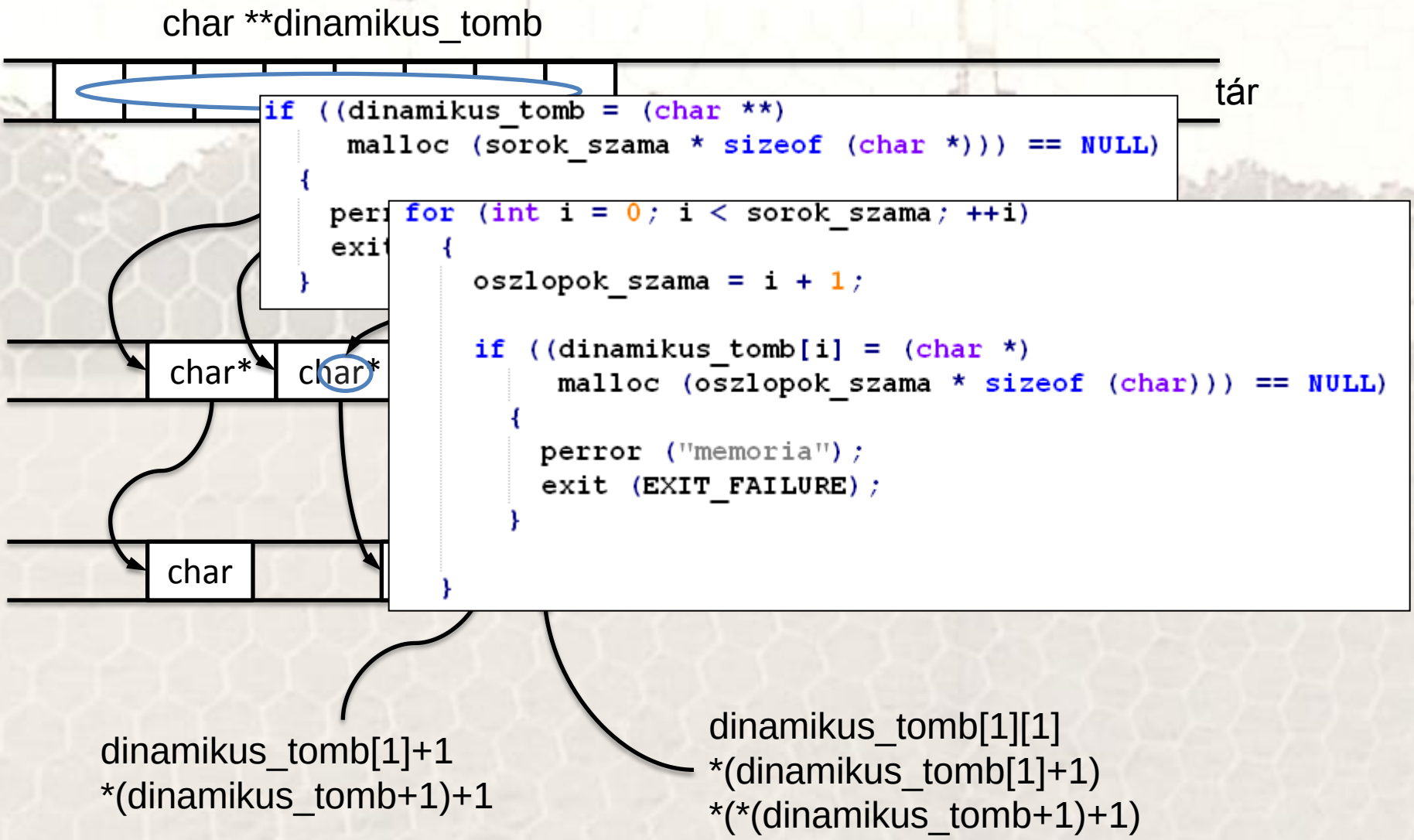
C és C++ összehasonlítása

Már szert tettünk annyi tapasztalatra, hogy megtegyük az első összehasonlításokat.

C	C++
Dinamikus tárkezelés	
malloc/free	new/delete

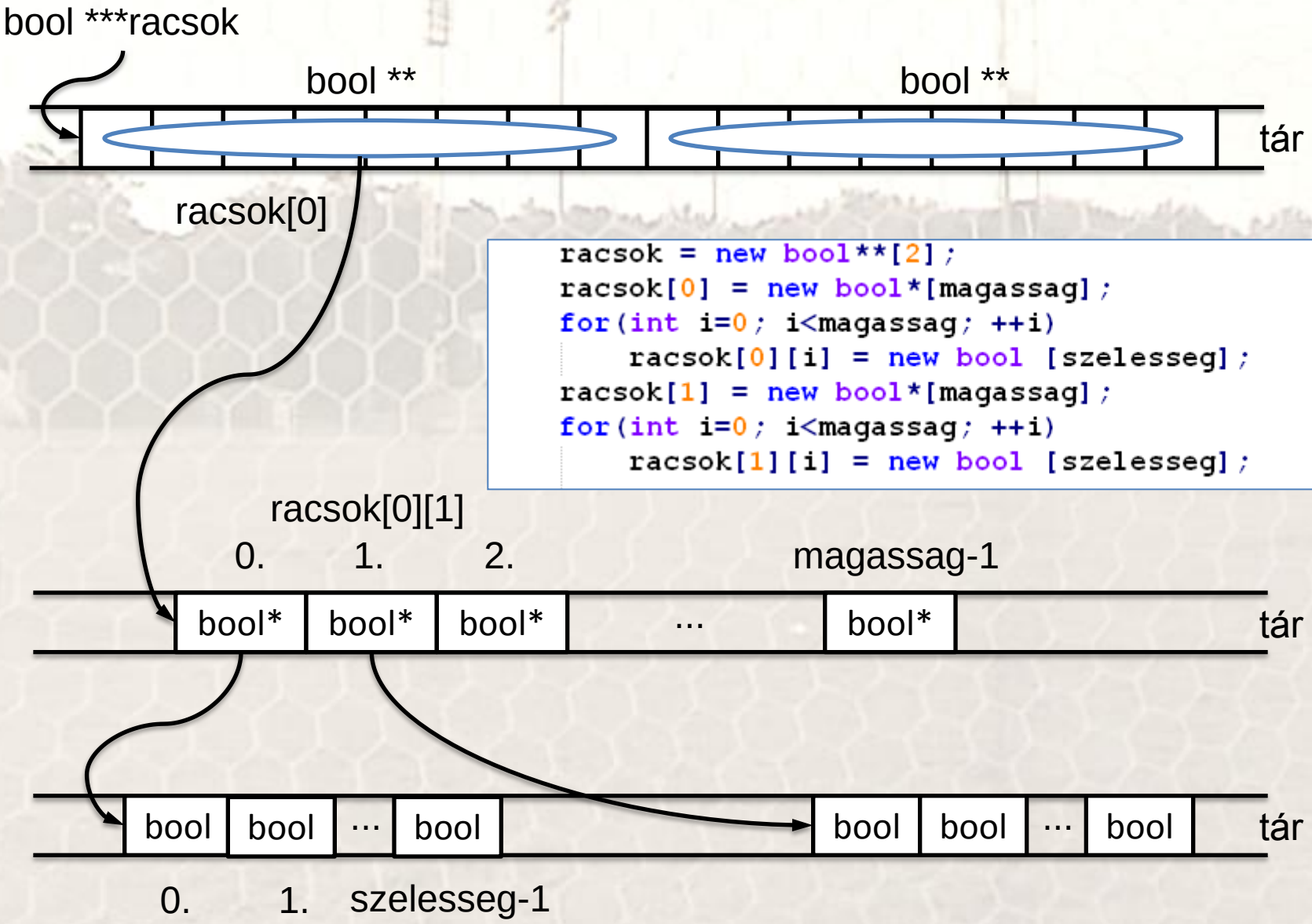
C

Ismétlés (2. ea.): Mutatók és többdimenziós tömbök



C++

Ismétlés (5. ea.): a Conway-féle életjátékunkból
Dinamikus memóriakezelés (malloc()-os ismétlő ábra)




```
racsok = new bool**[2];
racsok[0] = new bool*[magassag];
for(int i=0; i<magassag; ++i)
    racsok[0][i] = new bool [szelesseg];
racsok[1] = new bool*[magassag];
for(int i=0; i<magassag; ++i)
    racsok[1][i] = new bool [szelesseg];
```

C és C++ összehasonlítása

Már szert tettünk annyi tapasztalatra, hogy megtegyük az első összehasonlításokat.

C

Típusok

C++

referencia

```c
#include <stdio.h>

void
csere (int *a, int *b)
{
  *a = *a - *b;
  *b = *a + *b;
  *a = *b - *a;
}

int
main (void)
{
  int a = 5, b = 7;

  csere (&a, &b);
  printf ("a=%d b=%d\n", a, b);

  return 0;
}
```

Paraméterátadás

```c
#include <stdio.h>

void
csere (int &a, int &b)
{
  a = a - b;
  b = a + b;
  a = b - a;
}

int
main (void)
{
  int a = 5, b = 7;

  csere (a, b);
  printf ("a=%d b=%d\n", a, b);

  return 0;
}
```

referencia szerint

* const *

Már szert tettünk annyi tapasztalatra, hogy megtegyük az első összehasonlításokat.


```
double * const dp = &d;
*dp = -*dp;
```


```
const double * dp = &d;
*dp = -*dp;
```

T100: const double * const dp = &d;

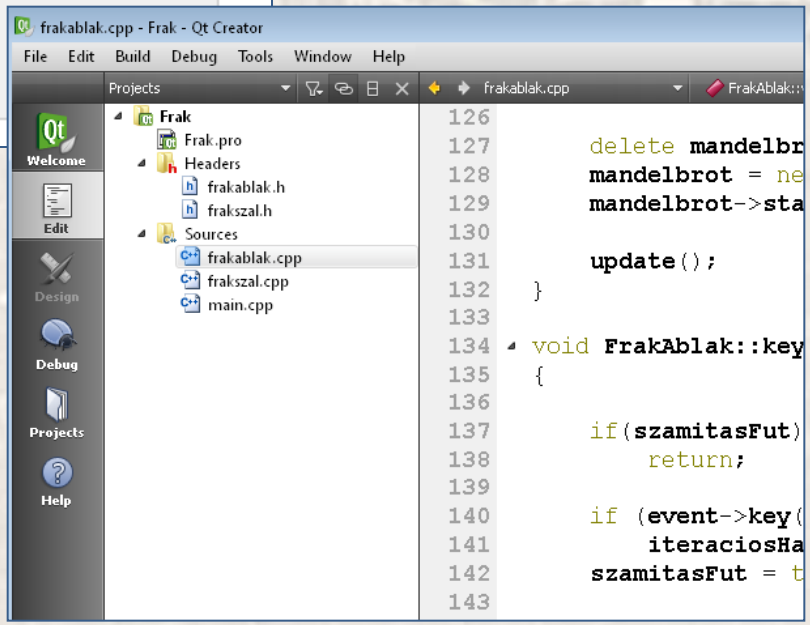
Labor

Qt

http://progpater.blog.hu/2011/02/26/tan_csodallak_amde_nem_ertelek_de_kepzetem_hegyvolgyedet_bejarja

Parancssorból így építheted, futtathatod:

```
[norbi@sgu Frak]$ ls -l
összesen 28
-rw-rw-r--. 1 norbi norbi 2917 febr  27 13.46 frakablak.cpp
-rw-rw-r--. 1 norbi norbi  626 febr  27 13.53 frakablak.h
-rw-rw-r--. 1 norbi norbi 4750 febr  27 13.46 frakszal.cpp
-rw-rw-r--. 1 norbi norbi  799 febr  27 13.54 frakszal.h
-rw-rw-r--. 1 norbi norbi 1408 febr  27 13.46 main.cpp
[norbi@sgu Frak]$ qmake-qt4 -project
[norbi@sgu Frak]$ qmake-qt4 Frak.pro
[norbi@sgu Frak]$ make
[norbi@sgu Frak]$./Frak
```

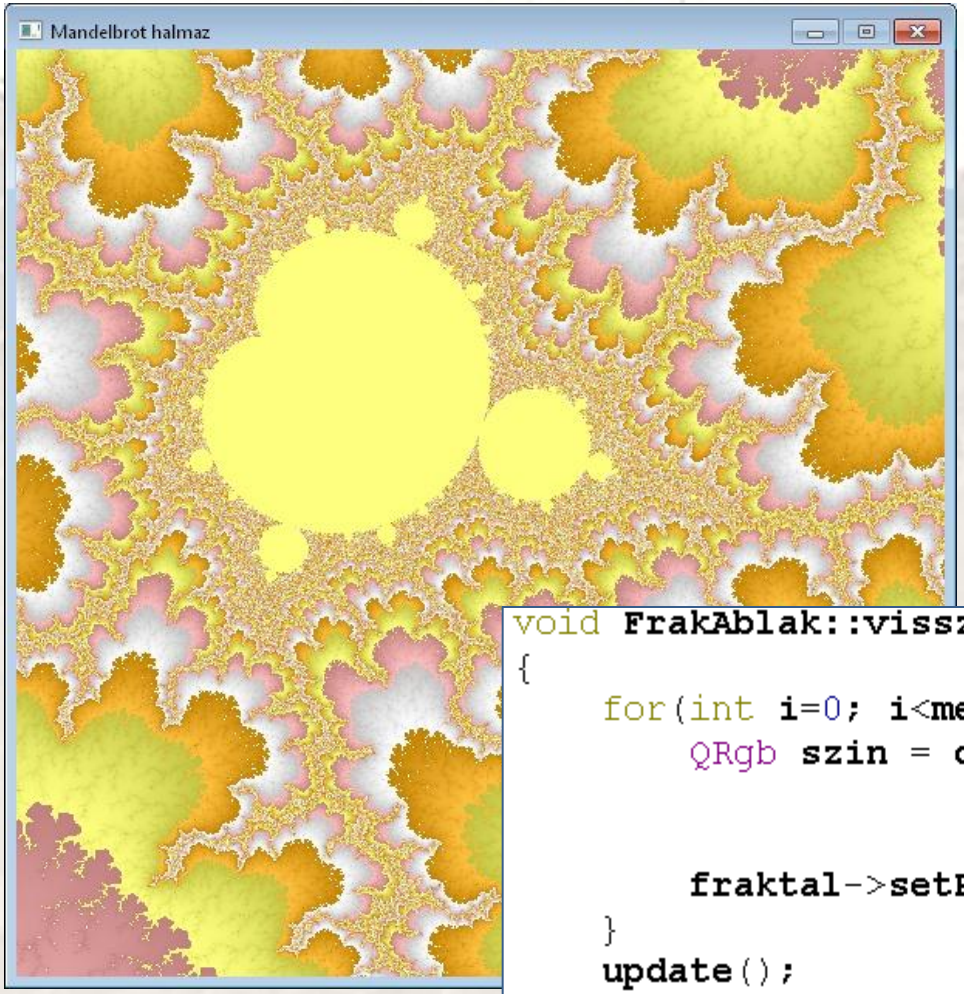


http://qt.nokia.com/downloads/

Labor

Qt

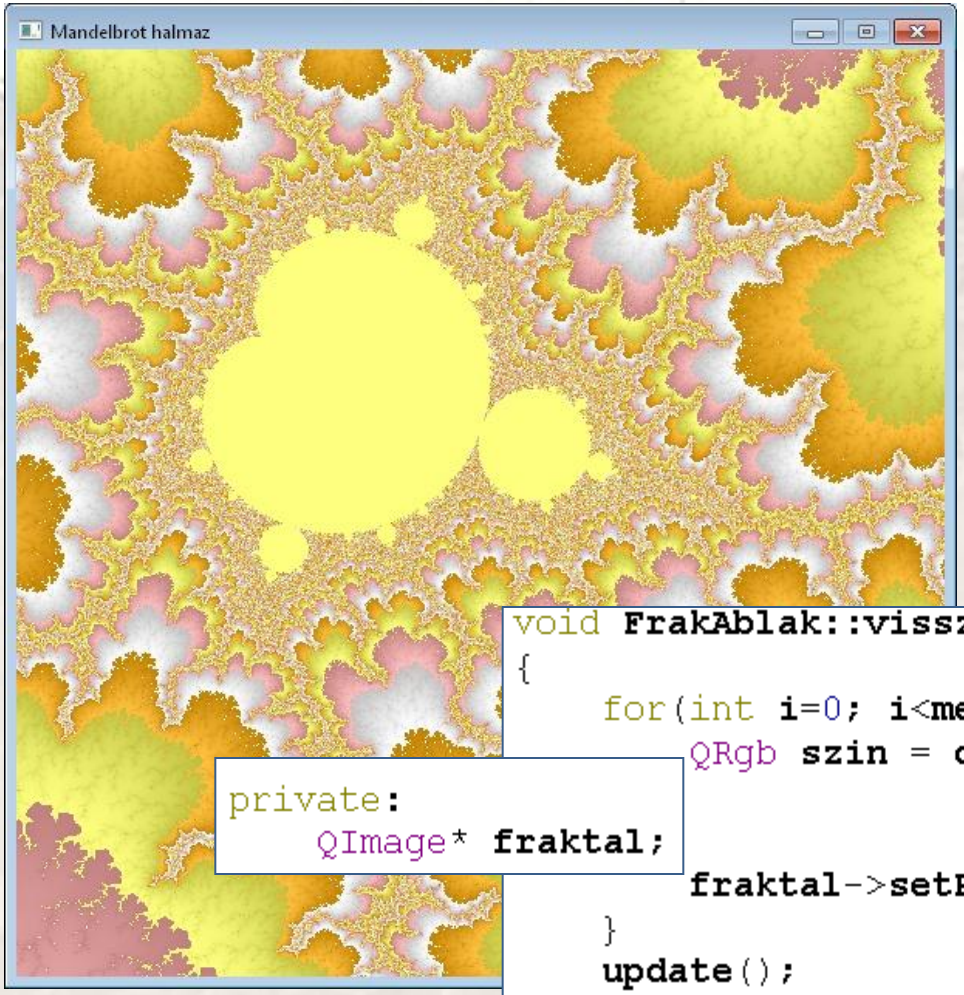
http://progpater.blog.hu/2011/02/26/tan_csodallak_amde_nem_ertelek_de_kepzetem_hegyvolgyedet_bejarja




```
void FrakAblak::vissza(int magassag, int *sor, int meret)
{
    for(int i=0; i<meret; ++i) {
        QRgb szin = qRgb(255-sor[i]%64,
                         255-sor[i]%128,
                         255-sor[i] );
        fraktal->setPixel(i, magassag, szin);
    }
    update();
}
```

Qt API doksi

http://doc.qt.nokia.com/4.7-snapshot/index.html

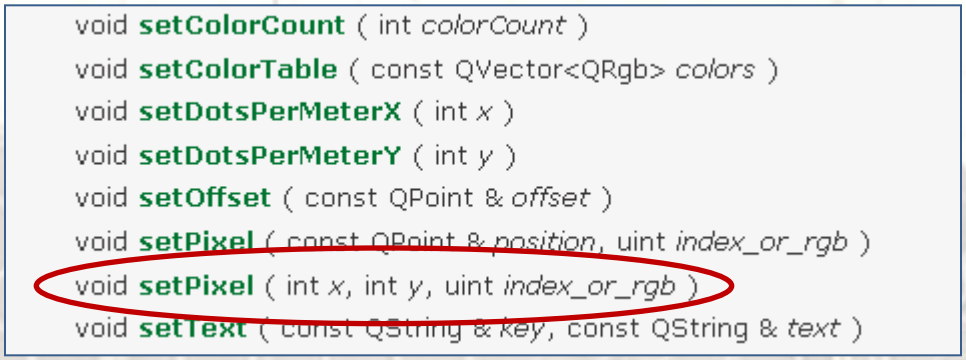



```
private:
    QImage* fraktal;
```

```
void FrakAblak::vissza(int magassag, int *sor, int meret)
{
    for(int i=0; i<meret; ++i) {
        QRgb szin = qRgb(255-sor[i]%64,
                         255-sor[i]%128,
                         255-sor[i] );
        fraktal->setPixel(i, magassag, szin);
    }
    update();
}
```

Qt API doksi

http://doc.qt.nokia.com/4.7-snapshot/index.html



http://doc.qt.nokia.com/4.7-snapshot/qimage.html

Akár magyarul is!

http://developer.qt.nokia.com/wiki/Roevid_utmutato_a_Qt_programozashoz

http://developer.qt.nokia.com/wiki/Getting_Started_Programming_with_QML_in_Hungarian

Rajzolás

```
void FrakAblak::paintEvent(QPaintEvent*) {
    QPainter qpainter(this);
    qpainter.drawImage(0, 0, *fraktal);
```

Qt Reference Documentation

Qt 4.7

Search index:

API Lookup

- Class index
- Function index
- Modules
- Namespaces
- Global Declaratio
- QML elements

Qt Topics

- Programming wit Qt
- Device UIs & Qt Quick
- UI Design with Q
- Cross-platform a Platform-specific
- Platform-specific info

Detailed Description

The QPainter class performs low-level painting on widgets and other paint devices.

QPainter provides highly optimized functions to do most of the drawing GUI programs require. It can draw ever chords. It can also draw aligned text and pixmaps. Normally, it draws in a "natural" coordinate system, but it c operate on any object that inherits the QPaintDevice class.

The common use of QPainter is inside a widget's paint event: Construct and customize (e.g. set the pen or the QPainter object after drawing. For example:

```
void SimpleExampleWidget::paintEvent(QPaintEvent *)
{
    QPainter painter(this);
    painter.setPen(Qt::blue);
    painter.setFont(QFont("Arial", 30));
    painter.drawText(rect(), Qt::AlignCenter, "Qt");
}
```

The core functionality of QPainter is drawing, but the class also provide several functions that allows you to c others that enable clipping. In addition you can control how different shapes are merged together by specifyin

The isActive() function indicates whether the painter is active. A painter is activated by the begin() function a The end() function, and the destructor, deactivates it.

Together with the QPaintDevice and QPaintEngine classes, QPainter form the basis for Qt's paint system. QPai

Egér és billentyű események

```
void FrakAblak::paintEvent(QPaintEvent*) {
    QPainter qpainter(this);
    qpainter.drawImage(0, 0, *fraktal);
    if(!szamitasFut) {
        qpainter.setPen(QPen(Qt::white, 1));
        qpainter.drawRect(x, y, mx, my);
    }
    qpainter.end();
}
```

```
protected:
    void paintEvent(QPaintEvent*);
    void mousePressEvent(QMouseEvent*);
    void mouseMoveEvent(QMouseEvent*);
    void mouseReleaseEvent(QMouseEvent*);
    void keyPressEvent(QKeyEvent*);
```

```
void FrakAblak::mousePressEvent(QMouseEvent* event) {

    // A nagyítandó kijelölt területet bal felső sarka:
    x = event->x();
    y = event->y();
    mx = 0;
    my = 0;

    update();
}
```

Egér és billentyű események

```
void FrakAblak::paintEvent(QPaintEvent*) {
    QPainter qpainter(this);
    qpainter.drawImage(0, 0, *fraktal);
    if(!szamitasFut) {
        qpainter.setPen(QPen(Qt::white, 1));
        qpainter.drawRect(x, y, mx, my);
    }
    qpainter.end();
}
```

```
void FrakAblak::mouseMoveEvent(QMouseEvent* event) {

    // A nagyítandó kijelölt terület szélessége és magassága:
    mx = event->x() - x;
    my = mx; // négyzet alakú

    update();
}
```

Egér és billentyű események

```
void FrakAblak::mouseReleaseEvent(QMouseEvent* event) {

    if(szamitasFut)
        return;

    szamitasFut = true;

    double dx = (b-a)/szelesseg;
    double dy = (d-c)/magassag;

    double a = this->a+x*dx;
    double b = this->a+x*dx+mx*dx;
    double c = this->d-y*dy-my*dy;
    double d = this->d-y*dy;

    this->a = a;
    this->b = b;
    this->c = c;
    this->d = d;

    delete mandelbrot;
    mandelbrot = new FrakSzal(a, b, c, d, szelesseg, magassag, iteraciosHatar, this);
    mandelbrot->start();

    update();
}
```

Egér és billentyű események

```
void FrakAblak::keyPressEvent(QKeyEvent *event)
{

    if(szamitasFut)
        return;

    if (event->key() == Qt::Key_N)
        iteraciosHatar *= 2;
    szamitasFut = true;

    delete mandelbrot;
    mandelbrot = new FrakSzal(a, b, c, d, szelesseg, magassag, iteraciosHatar, this);
    mandelbrot->start();

}
```

Mandelbrot nagyítása

```
void FrakAblak::paintEve
    QPainter qpainter(th
    qpainter.drawImage(0
    if(!szamitasFut) {
        qpainter.setPen(
        qpainter.drawRec
    }
    qpainter.end();
}
```

Labor

Visszatekintés: a hálózati ló

http://progpater.blog.hu/2011/03/06/halozati_vegyertek

```
if ((szemafor =
            semget (ftok (".", 42), 1, IPC_CREAT | S_IRUSR | S_IWUSR)) == -1)
{
    perror ("semget");
    exit (EXIT_FAILURE);
}
printf ("szemafor: %d\n", szemafor);
fflush (stdout);
if (semctl (szemafor, 0, SETVAL, 1))
{
    perror ("semctl");
    exit (EXIT_FAILURE);
}
```

Labor

Visszatekintés: a hálózati ló

http://progpater.blog.hu/2011/03/06/halozati_vegyertek

```
struct sembuf zar, nyit;
zar.sem_num = 0;
zar.sem_op = -1;
nyit.sem_num = 0;
nyit.sem_op = 1;

int olvasott = read (kliens, buffer2, 10);
write (kliens, buffer2, olvasott);

if (semop (szemafor, &zar, 1) == -1)
{
    perror ("semop");
    exit (EXIT_FAILURE);
}

++*(osztott_memoria_terulet+1);

if (buffer2[0] == '+')
    ++*osztott_memoria_terulet;
else
    --*osztott_memoria_terulet;
```

1

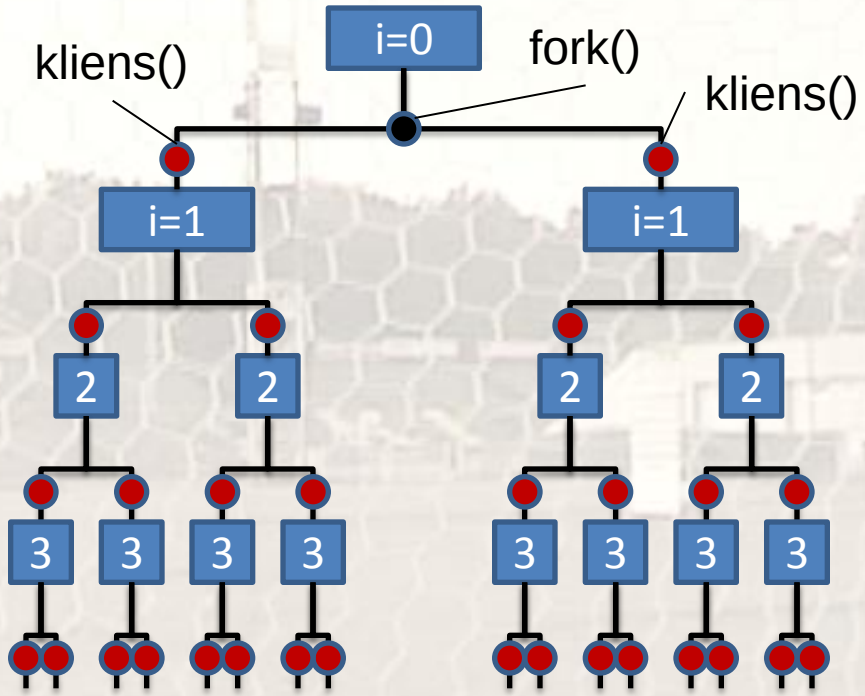
0

Labor

Visszatekintés: a hálózati ló, DOS

http://progpater.blog.hu/2011/03/18/a_hatodik_hetedik_labor

```
int
main (int argc, char *argv[])
{
    int gyermekem_pid;
    int statusz;

    for (int i=0; i<4; ++i)
        if ((gyermekem_pid = fork()) == 0)
        {
            kliens(argv[1][0]);
        }
        else if (gyermekem_pid > 0)
        {
        kliens(argv[1][0]);
            wait(&statusz);
        }
        else
        {
            exit(-1);
        }
    return 0;

    exit (EXIT_SUCCESS);
}
```



BB

```
#define NOF_FROM 4
#define NOF_TO 12

/* (state, read) -> */
int from[NOF_FROM][2] = {
  {0, 0},
  {0, 1},
  {1, 0},
  {1, 1}
};

/* -> (state, write, move) */
int to[NOF_TO][3] = {
  {0, 0, 0},
  {0, 0, 1},
  {0, 0, 2},
  {0, 1, 0},
  {0, 1, 1},
  {0, 1, 2},
  {1, 0, 0},
  {1, 0, 1},
  {1, 0, 2},
  {1, 1, 0},
  {1, 1, 1},
  {1, 1, 2}
};
```

Részletek: 1.,2. előadás és http://arxiv.org/abs/0908.1159

BB

```c
/* A T. machine */
int machine[NOF_FROM][5];

/* The Tape of the Machine */
#define NOF_CELLS 4096
int tape[NOF_CELLS];
int tapei;

/* The State of the Machine */
int state;

#define MAX_STEPS 1000

int
nof_ones ()
{
  int nof = 0, i;

  for (i = 0; i < NOF_CELLS; ++i)
    if (tape[i] == 1)
      ++nof;

  return nof;
}
```

Részletek: 1.,2. előadás és http://arxiv.org/abs/0908.1159

BB

```
/* Simulates Machines which are containing n rule(s) */
int
simulate (int nof, ...)
{
  va_list vap;
  int i, f, t, s, r;

  va_start (vap, nof);

  for (i = 0; i < nof; ++i)
    {
      f = va_arg (vap, int);
      t = va_arg (vap, int);

      machine[i][0] = from[f][0];
      machine[i][1] = from[f][1];
      machine[i][2] = to[t][0];
      machine[i][3] = to[t][1];
      machine[i][4] = to[t][2];

    }

  va_end (vap);
```

Részletek: 1.,2. előadás és http://arxiv.org/abs/0908.1159

BB

```
for (i = 0; i < NOF_CELLS; ++i)
  tape[i] = 0;

tapei = NOF_CELLS / 2;
state = 0;

for (s = 0; s < MAX_STEPS; ++s)
  {
    r = -1;
    for (i = 0; i < nof; ++i)
      {
        if (machine[i][0] == state && machine[i][1] == tape[tapei])
          {
            r = i;
            break;
          }
      }

    if (r == -1)                //halt
      return nof_ones ();
    else
      {
        state = machine[r][2];
        tape[tapei] = machine[r][3];
        tapei += (machine[r][4] - 1);
      }
  }

return -1;
}
```

Részletek: 1.,2. előadás és http://arxiv.org/abs/0908.1159

BB

```
// Simulate Machines containing 1 rule
// Simulate Machines containing 2 rules
// Simulate Machines containing 3 rules
// Simulate Machines containing 4 rules
for (i = 0; i < NOF_FROM; ++i)
  for (j = i+1; j < NOF_FROM; ++j)
    for (k = j+1; k < NOF_FROM; ++k)
      for (l = k+1; l < NOF_FROM; ++l)

        for (m = 0; m < NOF_TO; ++m)
          for (n = 0; n < NOF_TO; ++n)
            for (x = 0; x < NOF_TO; ++x)
              for (y = 0; y < NOF_TO; ++y)
                if ((h = simulate (4, i, m, j, n, k, x, l, y)) > max_ones)
                  {
                    max_ones = h;
                    printf ("Number of 1's is %d\n", max_ones);
                    print (4, i, m, j, n, k, x, l, y);
                  }

printf("The BB(2) Computing is over.");
return 0;
```

Részletek: 1.,2. előadás és http://arxiv.org/abs/0908.1159

BB

```
nbatfai@hpserver:~/BB/2> ./bb2
Number of 1's is 1
(0, 0)->(0, 1, 1)
Number of 1's is 2
(0, 0)->(1, 1, 0)
(1, 0)->(0, 1, 1)
Number of 1's is 3
(0, 0)->(1, 1, 0)
(0, 1)->(0, 1, 2)
(1, 0)->(0, 1, 1)
Number of 1's is 4
(0, 0)->(1, 1, 0)
(0, 1)->(1, 1, 2)
(1, 0)->(0, 1, 2)
The BB(2) Computing is over.nbatfai@hpserver:~/BB/2>
```

Részletek: 1.,2. előadás és http://arxiv.org/abs/0908.1159

Laborkártyák

Lefordul? Ha igen, mit ír ki? S miért?

```
#include <stdio.h>

int
main (void)
{

  int a = 5;
  int& ar = a;
  int& arr = ar;
  ++arr;

  printf ("a=%d\n", a);

  return 0;
}
```

Laborkártyák

Lefordul? Ha igen, mit ír ki?

```
#include <iostream>
#include <vector>

int
main (int argc, char *argv[])
{

  std::vector <char> v(3);

  v[0] = v[1] = v[2] ='a';

  std::cout << v[1] << std::endl;

  return 0;
}
```

Lefordul?

```
int
main (int argc, char *argv[])
{

  Verem v (512);

  v.push ('0');
  v.push ('1');

  std::cout << v[1] << std::endl;
  std::cout << v[0] << std::endl;

  return 0;
}
```

Laborkártyák

Lefordul, ha a jobb oldal is tagja a saját vermünknek?

```
int
main (int argc, char *argv[])
{

  Verem v (512);

  v.push ('0');
  v.push ('1');

  std::cout << v[1] << std::endl;
  std::cout << v[0] << std::endl;

  return 0;
}
```

```
char& operator[](int i){
  return verem[i];
}
```

Laborkártyák

Lefordul? Ha igen, mit ír ki?

```
#include <iostream>
#include <vector>

int* tomb(int meret) {

  return new int[meret];
}

int
main (int argc, char *argv[])
{

  int* t = tomb(10);

  t[6] = 42;

  std::cout << t[6] << std::endl;

  delete [] t;

  return 0;
}
```

Lefordul? Ha igen, mit ír ki?

```
#include <iostream>
#include <vector>

int* tomb(int meret) {

  int* t = new int[meret];
  return t;
}

int
main (int argc, char *argv[])
{

  int* t = tomb(10);

  t[6] = 42;

  std::cout << t[6] << std::endl;

  delete [] t;

  return 0;
}
```

Mi a különbség?

Laborkártyák

Mit mondanál erről a kódról?

```
#include <iostream>
#include <vector>

int* tomb(int meret) {

  int* t1 = new int[meret];
  int* t2 = new int[meret];
  return t2;
}

int
main (int argc, char *argv[])
{

  int* t = tomb(10);

  t[6] = 42;

  std::cout << t[6] << std::endl;

  delete [] t;

  return 0;
}
```

Mit mondanál erről a kódról?

```
#include <iostream>
#include <vector>

int* tomb(int meret) {

  int* t = new int[meret];
  return  new int[meret];
}

int
main (int argc, char *argv[])
{

  int* t = tomb(10);

  t[6] = 42;

  std::cout << t[6] << std::endl;

  delete [] t;

  return 0;
}
```

Mi a különbség?

Laborkártyák

Milyen a librcsc és az agent2D licence?

```
    rcsc::BasicClient client;

    if ( ! agent.init( &client, argc, argv ) )
    {
        return EXIT_FAILURE;
    }

    /*
      You should add your copyright message here.
     */
     std::cout << "**************************************************************\n"
               << " This program is modified by FerSML team\n"
               << " Copyright 2011. Norbert Bátfai.\n"
               << " University of Debrecen\n"
               << " All rights reserved.\n"
               << "**************************************************************\n"
               << std::flush;

    /*
      Do NOT remove the following copyright notice!
     */
    std::cout << "**************************************************************\n"
              << " This program is based on agent2d created by Hidehisa Akiyama.\n"
              << " Copyright 2006 - 2010. Hidehisa Akiyama.\n"
              << " National Institute of Advanced Industrial Science and Technology\n"
              << " All rights reserved.\n"
              << "**************************************************************\n"
              << std::flush;

    client.run( &agent );

    return EXIT_SUCCESS;
}
```

main_player.cpp

Laborkártyák

Mi itt a helyzet a másoló konstruktorral és az értékadó operátorral (másoló értékadással)?

```
class SoccerAgent {
public:
    friend class BasicClient;

protected:
    //! interface to the rcssserver
    BasicClient * M_client;

private:

    // nocopyable
    SoccerAgent( const SoccerAgent & );
    SoccerAgent & operator=( const SoccerAgent & );

public:

    /*!
      \brief nothing to do. just set NULL to M_client
     */
    SoccerAgent();
```

soccer_agent.h

Laborkártyák

Mit ír ki? (Ha lefordul.)

```
#include <iostream>

int
main (int argc, char *argv[])
{

  double d = 2.71;

  double *const dp1 = &d;
  *dp1 = -*dp1;

  std::cout << d << std::endl;

  const double *dp2 = &d;
  *dp2 = -*dp2;

  std::cout << d << std::endl;

  const double *const dp3 = &d;
  *dp3 = -*dp3;

  std::cout << d << std::endl;

  return 0;
}
```

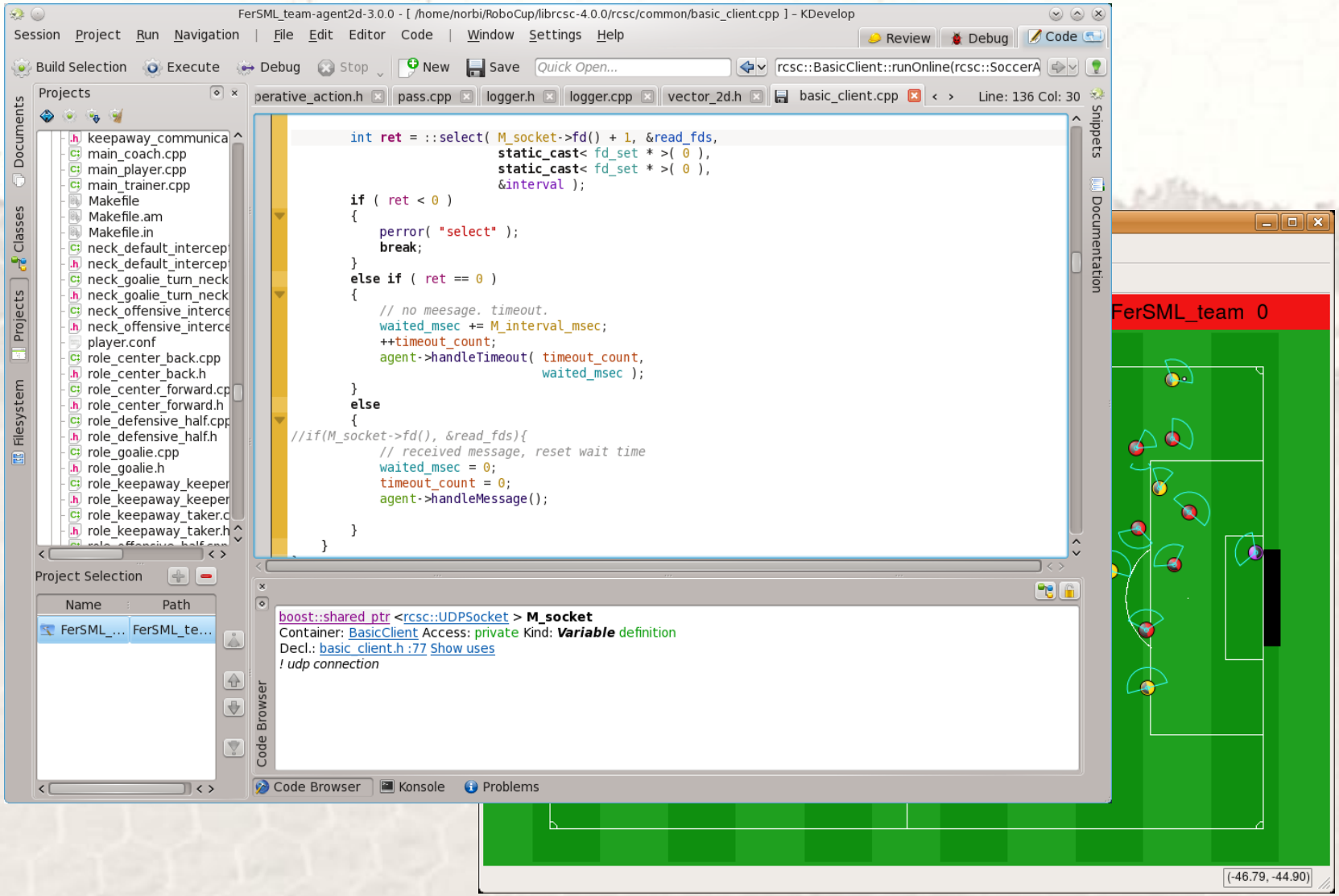
Ismétlő laborkártyák

Mi lesz a gyümölcskosárban?

```c
#include <stdio.h>
#include <string.h>

int
main (int argc, char **argv)
{
  char *s1 = "dio";
  char s2[] = "dio";
  char s3[] = { 'd', 'i', 'o', '\0' };

  if (s1 == s2)
    printf ("alma\n");
  if (!strcmp (s2, s3))
    printf ("korte\n");
  if (strcmp (s1, s3) == 0)
    printf ("banan\n");

  return 0;
}
```

Otthoni opcionális feladat

A robotfoci japán szoftvereinek (librcsc, agent2d) tanulmányozása a KDevelop-ban.



Otthoni opcionális feladat

A robotfoci japán szoftvereinek (librcsc, agent2d) tanulmányozása a KDevelop-ban.

Kötelező olvasmány

(B&L könyv)
Benedek Zoltán, Levendovszky Tihamér: Szoftverfejlesztés C++ nyelven, Budapest, 2007, Szak K.

17-58
73-91
93-110
(két hét múlvára)

(S könyv)
Stroustrup, Bjarne: A C++ programozási nyelv, Kiskapu, 2001.
http://webpac.lib.unideb.hu/WebPac/CorvinaWeb?action=onelong&showtype=longlong&recnum=255516&pos=3

295-330

„Nem tudok kimerítő leírást adni arról, hogy hogyan tudsz megtanulni programozni – nagyon összetett tudásról van szó. Egyet azonban elárulhatok: a könyvek és tanfolyamok nem érnek túl sokat (sok, valószínűleg a legtöbb hacker autodidakta). Aminek van értelme: (a) kódot olvasni és (b) kódot írni." - Eric Steven Raymond: How To Become A Hacker

A magyar fordítás: http://esr.fsf.hu/hacker-howto.html

Ajánlott olvasmány

(OSS könyv)
Alan Ezust, Paul Ezust: ***An Introduction to Design Patterns in C++ with Qt 4***, Prentice Hall (Open Source Series) 2006

Pdf-ben:
http://ptgmedia.pearsoncmg.com/images/9780131879058/downloads/0131879057_Ezust_book.pdf

47-79 (Hasonlóan hozzánk, itt is UML-el támogatott a tárgyalás.)

(24 könyv)
Jesse Liberty, Horvath, David B. Büki András: ***Tanuljuk meg a C++ programozási nyelvet 24 óra alatt***
http://webpac.lib.unideb.hu/WebPac/CorvinaWeb?action=onelong&showtype=longlong&recnum=469876&pos=2

131-159

„Nem tudok kimerítő leírást adni arról, hogy hogyan tudsz megtanulni programozni – nagyon összetett tudásról van szó. Egyet azonban elárulhatok: a könyvek és tanfolyamok nem érnek túl sokat (sok, valószínűleg a legtöbb hacker autodidakta). Aminek van értelme: (a) kódot olvasni és (b) kódot írni." - Eric Steven Raymond: How To Become A Hacker

A magyar fordítás: http://esr.fsf.hu/hacker-howto.html