Prog1, C++ a gyakorlatban

Magasszintű programozási nyelvek

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_8.ppt, v.: 0.0.3, 2011. 05. 15.
http://www.inf.unideb.hu/~nbatfai/#p1

Az óra blogja: http://progpater.blog.hu/

Felhasználási engedély

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

Célok és tartalom

Előadás

a) Funktorok
b) STL (a szabványos C++ sablonkönyvtár)
c) Tervezési minták (tartalmazás, hogyan?)

Labor

a) Boost

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/2011/04/10/a_kilencedik_tizedik_labor
http://progpater.blog.hu/2011/04/11/imadni_fogjak_a_c_t_egy_emberkent_tiszta_szivbol_3
http://progpater.blog.hu/2011/04/14/egyutt_tamadjuk_meg

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

1) A hallgató egyszerű példáiban tudja az STL szabványos tárolóit (vektor, lista, halmaz, asszociatív tömb, bithalmaz) használni.

Az svn tároló elérése kapcsán:
http://progpater.blog.hu/2011/02/05/az_elso_labor/fullcommentlist/1#c12856691

Minimális elméleti cél

1) Vektor (dinamikus tömb, vector)
2) Lista (kétszeresen láncolt, list)
3) Halmaz, set
4) Asszociatív tömb, map
5) Bithalmaz

Funktorok

A () fgv. Hívás operátort túlterhelő osztály (struktúra).

a) Funktor STL algoritmusban

Funktorok

Ism.: fgv. Hívás, paraméterátadás vermen

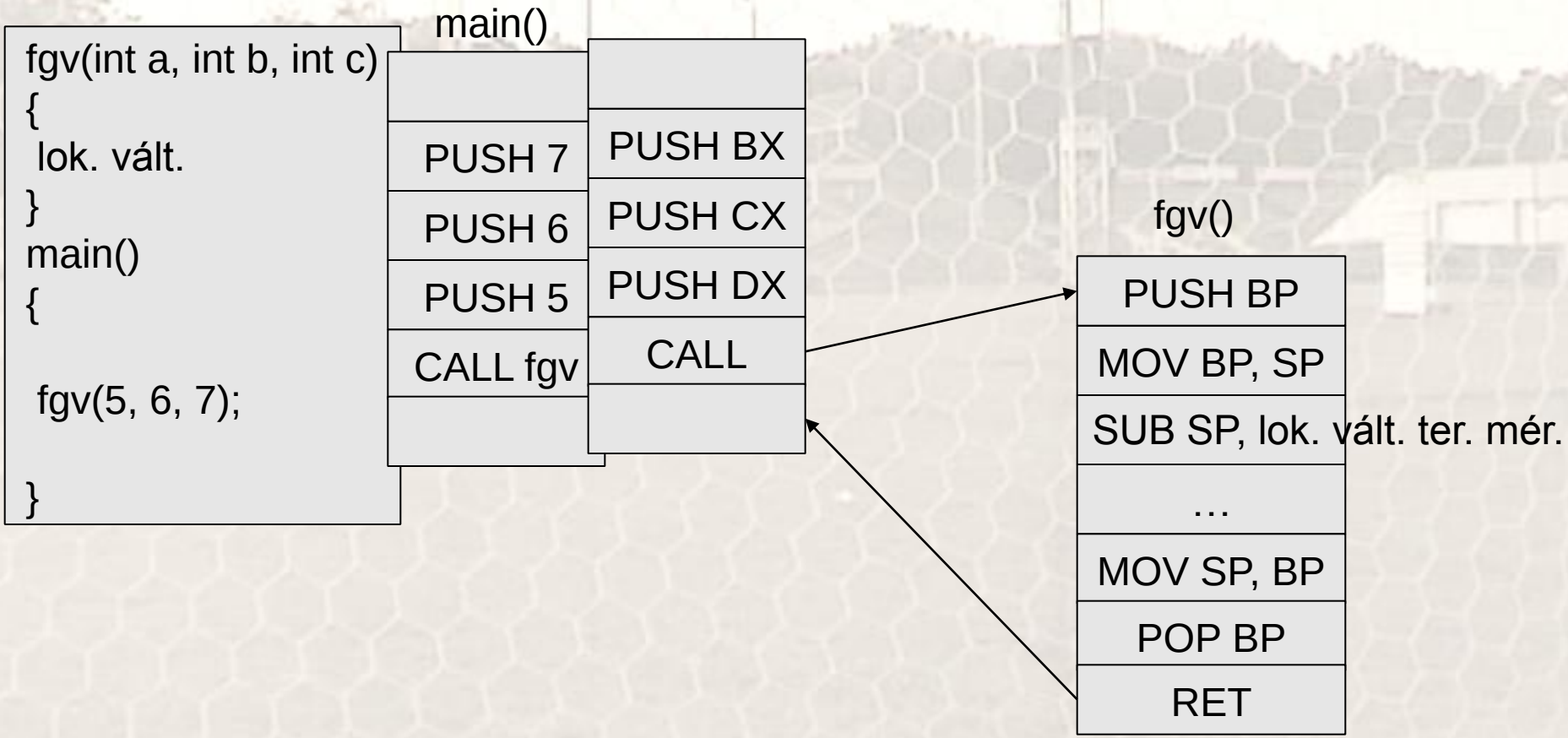

```
typedef void (*BADBOY)();

BADBOY badboy;

...

try_one_crash()
{if (nbytes > 0)
    (*badboy)();
 else if (nbytes == 0)
   while(1);}
```

Funktorok

Ism.: paraméterátadás

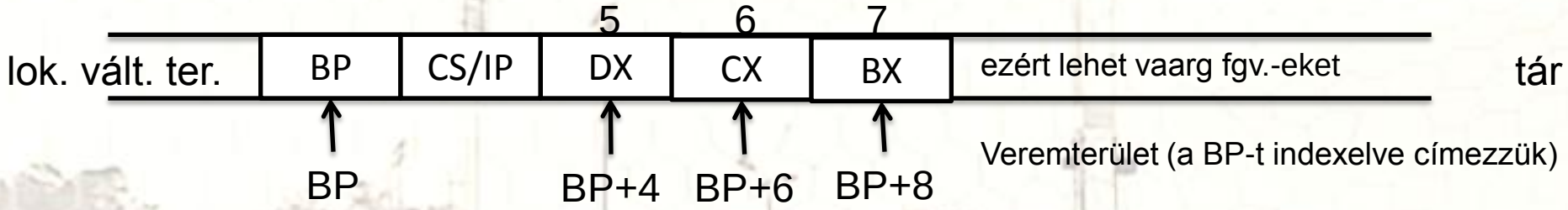
a) Regiszterek
b) Közös memória
c) Verem (tipikusan a magasszintű programozási nyelvek)

lsm: fgv. hívás, paraméterátadás, lokális változók



Ism.: fgv. hívás, paraméterátadás, lokális változók

Visszatérési cím mentése, pl. IP a köv. végrehajtandó offszetje




```
fgv(int a, int b, int c)
{
 lok. vált.
}
main()
{

 fgv(5, 6, 7);

}
```

main()

PUSH 7	PUSH BX
PUSH 6	PUSH CX
PUSH 5	PUSH DX
CALL fgv	CALL
	ADD SP, 3*(2)

fgv()

PUSH BP
MOV BP, SP
SUB SP, lok. vált. ter. mér.
…
MOV SP, BP
POP BP
RET

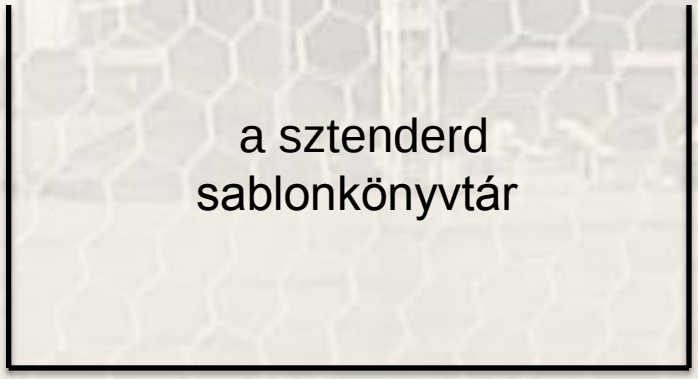
Funktorok

```
typedef std::bitset<50> TudatMinta;

struct tudatRendezes
{
    bool operator()(const TudatMinta& a, const TudatMinta& b) {
        LZWBinFa afa, bfa;
        for (int i=0; i<a.size(), i<b.size(); ++i) {
            afa << a[i];
            bfa << b[i];
        }
        return bfa.getSzoras() < afa.getSzoras();
    }
};

int
main (int argc, char *argv[])
{
    std::vector< TudatMinta > v;

    TudatMinta elso(std::string("11111111111111111111111111111111111111111111111111"));
    TudatMinta masodik(std::string("11111000001111100000111110000011111000001111100000"));
    TudatMinta harmadik(std::string("01100101100111111001001101100110000110011001010110111"));
    v.push_back(elso);
    v.push_back(masodik);
    v.push_back(harmadik);

    std::copy(v.begin(), v.end(), std::ostream_iterator<TudatMinta>(std::cout, "\n"));
    std::sort(v.begin(), v.end(),  tudatRendezes());
    std::copy(v.begin(), v.end(), std::ostream_iterator<TudatMinta>(std::cout, "\n"));
}
```

STL

Standard Template Library
(a szabványos C++ sablonkönyvtár)

a) C programozóknak: a sztenderd függvénykönyvtár
(pl.:GNU C Library, glibc, http://www.gnu.org/s/libc/)
b) C++ programozóknak: a sztenderd osztálykönyvtár
(pl.: GNU Standard C++ Library, libstdc++, http://gcc.gnu.org/libstdc++/)



A GNU-s sztenderd osztálykönyvtár API doksija:
http://gcc.gnu.org/onlinedocs/libstdc++/api.html

Ism.: Helló, Világ! - .S

Rendszerhívással

```
.data

hello:
        .ascii  "Hello, Vilag!"

.text
        .global _start

_start:
        movl $4, %eax
        movl $1, %ebx
        movl $hello, %ecx
        movl $14, %edx

        int $0x80

        movl $1, %eax
        movl $0, %ebx

        int $0x80
```

A megfelelő regisztekebe töltjük a rendszerhívás kódját és paramétereit:

```
write (1, "Hello, Vilag!", 14);
 EAX  EBX       ECX         EDX
```

```
exit (0);
 EAX EBX
```

```
$ as asmhello.S -o asmhello.o
$ ld asmhello.o -o asmhello
$ ./asmhello
Hello, Vilag!
```

Ism.: Helló, Világ! - write

```
#include <unistd.h>

int
main ()
{

  write (1, "Hello, Vilag!", 14);

  return 0;
}
```

Rendszerhívással

```
$ gcc writehello.c -o writehello
$ ./writehello
Hello, Vilag!
```

```
WRITE(2)                  Linux Programmer's Manual                 WRITE(2)

NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);

DESCRIPTION
...
```

Rendszerhívással

(Az előző fólia példában a printf() hívja a write()-ot.)

Ism.: Helló, Világ! - printf

```
#include <stdio.h>

int
main ()
{

  printf ("Hello, Vilag!");

  return 0;
}
```

Könyvtári függvénnyel

```
$ gcc printfhello.c -o printfhello
$ ./printfhello
Hello, Vilag!
```

```
PRINTF(3)                 Linux Programmer's Manual                 PRINTF(3)

NAME
       printf,  fprintf,  sprintf,  snprintf,  vprintf, vfprintf, vsprintf,
       vsnprintf - formatted output conversion

SYNOPSIS
       #include <stdio.h>

       int printf(const char *format, ...);
...
```

Könyvtári függvénnyel

Ism.: Glibc trófeák

Kisbajnokság: definiáld felül a glibc egy (változó argumentumszámú) függvényét!

```
$ gcc -shared -Wl,-soname,libsajat.so.1 -o libsajat.so.1.0 libsajat.c
$ ln -s libsajat.so.1.0 libsajat.so.1
$ ln -s libsajat.so.1 libsajat.so
$ export LD_PRELOAD=libsajat.so
$ export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH
$ ./teszt
H 1 e 2 l 3 l 4 o
$
```

Ha gond lenne a fordítással:
http://tldp.fsf.hu/HOWTO/Program-Library-HOWTO-hu/shared-libraries.html

Ism.: Glibc trófeák

libsajat.c

```
#include <errno.h>
#include <stdarg.h>

int
printf (char *fmt, ...)
{
  va_list ap;
  errno = 1;

  va_start (ap, fmt);
  putchar (0x1B);
  putchar ('[');
  putchar ('4');
  putchar ('7');
  putchar (';');
  putchar ('3');
  putchar ('1');
  putchar ('m');
  vprintf (fmt, ap);
  putchar (0x1B);
  putchar ('[');
  putchar ('0');
  putchar ('m');
  va_end (ap);

  return 0;
}
```

teszt.c

```
#include <stdio.h>

int
main (void)
{
  printf ("H %d e %d l %d l %d o\n", 1, 2, 3, 4);
}
```

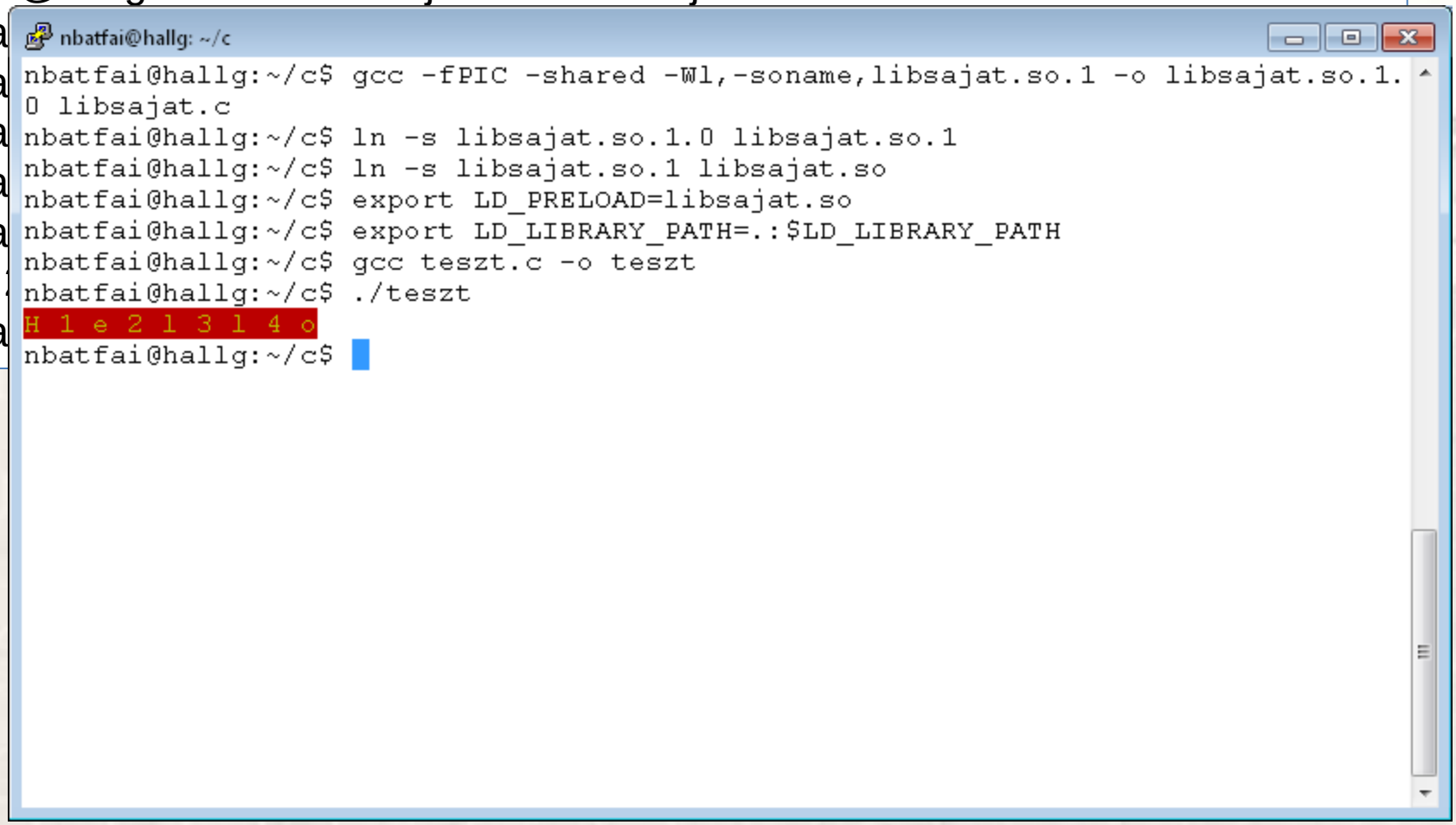
háttér- és szövegszín

ANSI Escape szekvenciák
http://en.wikipedia.org/wiki/ANSI_escape_code

visszaállítás

lsm.: Glibc trófeák

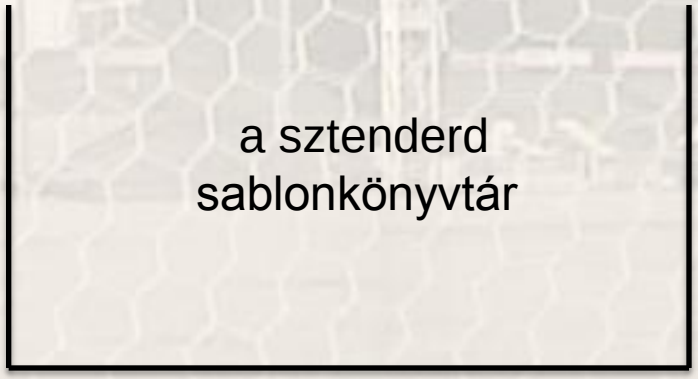
```
nbatfai@hallg:~/c$ gcc -fPIC -shared -Wl,-soname,libsajat.so.1 -o libsajat.so.1.0
libsajat.c
nbatfai@hallg:~/c$ ln -s libsajat.so.1.0 libsajat.so.1
nbatfa
nbatfa
nbatfa
nbatfa
nbatfa
H 1 e
nbatfa
```



STL

Standard Template Library
(a szabványos C++ sablonkönyvtár)

a) C programozóknak: a sztenderd függvénykönyvtár
(pl.:GNU C Library, glibc, http://www.gnu.org/s/libc/)
b) C++ programozóknak: a sztenderd osztálykönyvtár
(pl.: GNU Standard C++ Library, libstdc++, http://gcc.gnu.org/libstdc++/)



A GNU-s sztenderd osztálykönyvtár API doksija:
http://gcc.gnu.org/onlinedocs/libstdc++/api.html

API doksi

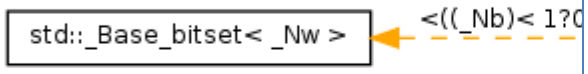
http://gcc.gnu.org/onlinedocs/libstdc++/api.html

std > std::bitset< _Nb >

std::bitset< _Nb > Class Template Reference [Containers]

The bitset class represents a *fixed-size* sequence of bits. More...

Inheritance diagram for std::bitset< _Nb >:



List of all members.

Classes

- class **reference**

Public Member Functions

- **bitset** ()
- **bitset** (unsigned long long __val)
- template<class _CharT , class _Traits
- template<class _CharT , class _Traits
- template<class _CharT , class _Traits
- **bitset** (const char *__str)
- size_t **_Find_first** () const

```
#include <bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
    long          a = 'a';
    bitset<10>    b(a);

    cout << "b('a') is " << b << endl;

    ostringstream s;
    s << b;
    string  str = s.str();
    cout << "index 3 in the string is " << str[3] << " but\n"
         << "index 3 in the bitset is " << b[3] << endl;
}
```

http://gcc.gnu.org/onlinedocs/libstdc++/libstdc++-api-4.5/a00396.html

Iterátor

végigvezetett mutató

```
#include <iostream>

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

  char c_string[] = "Hello, Vilag!";

  for (char *iter = c_string; *iter != 0; ++iter)
    std::cout << *iter;

  std::cout << std::endl;

  return 0;
}
```

Iterátor

végigvezetett mutató

```
#include <iostream>

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

  int egesz_tomb[6] = {0, 1, 2, 3, 4, 5};
  int* vege_utan = egesz_tomb + sizeof(egesz_tomb)/sizeof(int);

  for (int *iter = egesz_tomb; iter != vege_utan; ++iter)
    std::cout << *iter << ", ";

  std::cout << std::endl;

  return 0;
}
```

Iterátor

végigvezetett mutató

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

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

  std::vector < char >vektor;

  vektor.reserve(4);

  vektor.push_back ('h');
  vektor.push_back ('e');
  vektor.push_back ('l');
  vektor.push_back ('o');

  for (std::vector < char >::iterator iter = vektor.begin ();
       iter != vektor.end (); ++iter)
    std::cout << *iter << ", ";

  std::cout << std::endl;

  return 0;
}
```

Iterátor

végigvezetett mutató

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

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

  std::vector < int >vektor;

  vektor.reserve(4);

  vektor.push_back (1);
  vektor.push_back (2);
  vektor.push_back (41);
  vektor.push_back (42);

  for (std::vector < int >::iterator iter = vektor.begin ();
       iter != vektor.end (); ++iter)
    std::cout << *iter << ", ";

  std::cout << std::endl;

  return 0;
}
```

Iterátor

végigvezetett mutató

```
#include <iostream>
#include <vector>
#include "int.h"

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

  std::vector < Int >vektor;

  vektor.reserve(4);

  vektor.push_back (1);
  vektor.push_back (2);
  vektor.push_back (41);
  vektor.push_back (42);

  for (std::vector < Int >::iterator iter = vektor.begin ();
       iter != vektor.end (); ++iter)
    std::cout << *iter << ", ";

  std::cout << std::endl;

  return 0;
}
```

Elem hozzáadása

```
#include <iostream>
#include <vector>
#include "int.h"
int
main (int argc, char *argv[])
{
  std::vector < Int >vektor;

  vektor.reserve(4);

  Int egy(1);

  std::cout << "vektor.push_back (egy);" << std::endl << std::flush;
  vektor.push_back (egy);
  std::cout << "vektor.push_back (2);" << std::endl << std::flush;
  vektor.push_back (2);
  std::cout << "vektor.push_back (41);" << std::endl << std::flush;
  vektor.push_back (41);
  std::cout << "vektor.push_back (42);" << std::endl << std::flush;
  vektor.push_back (42);

  for (std::vector < Int >::iterator iter = vektor.begin ();
       iter != vektor.end (); ++iter)
    std::cout << *iter << ", ";

  std::cout << std::endl;
  return 0;
}
```

Elem hozzáadása

„laborkártyás logolásunkat” használva:

```
nbatfai@hallg:~$ g++ iter5.cpp int.cpp -o iter
nbatfai@hallg:~$ ./iter
-- Int ctor 0x7fffffffdf80
vektor.push_back (egy);
-- Int masolo ctor 0x604010
vektor.push_back (2);
-- Int ctor 0x7fffffffdfb0
-- Int masolo ctor 0x604014
-- Int dtor 0x7fffffffdfb0
vektor.push_back (41);
-- Int ctor 0x7fffffffdfc0
-- Int masolo ctor 0x604018
-- Int dtor 0x7fffffffdfc0
vektor.push_back (42);
-- Int ctor 0x7fffffffdfd0
-- Int masolo ctor 0x60401c
-- Int dtor 0x7fffffffdfd0
1, 2, 41, 42,
-- Int dtor 0x7fffffffdf80
-- Int dtor 0x604010
-- Int dtor 0x604014
-- Int dtor 0x604018
-- Int dtor 0x60401c
```

Ism.: egy asszociatív tárolós alternatíva

http://progpater.blog.hu/2011/04/03/elmondtam_milliomezerszer_2

```
#include <iostream>
#include <string>
#include <map>

int
main ()
{
    std::string s;
    std::map<std::string, int> szoszmap;

    while (std::cin >> s)
        ++szoszmap[s];

    for (std::map<std::string, int>::iterator iter = szoszmap.begin();
            iter != szoszmap.end(); ++iter)
        std::cout << iter->first << ", " << iter->second << std::endl;

    return 0;
}
```

Sablonparaméterek: <kulcs, érték>

```
nbatfai@hallg:~$ g++ szomap.cpp -o szomap
nbatfai@hallg:~$ ./szomap
alma kotre alma alma korte banan dio alma
alma, 4
banan, 1
dio, 1
korte, 1
kotre, 1
```

Ism.: egy asszociatív tárolós alternatíva

http://progpater.blog.hu/2011/04/03/elmondtam_milliomezerszer_2

```
#include <iostream>
#include <string>
#include <map>

int
main ()
{
    std::string s;
    std::map<std::string, int> szoszmap;

    while (std::cin >> s)
        ++szoszmap[s];

    for (std::map<std::string, int>::iterator iter = szoszmap.begin();
         iter != szoszmap.end(); ++iter)
        std::cout << iter->first << ", " << iter->second << std::endl;

    return 0;
}
```

iterátort a tároló első elemére

iterátort a tároló utolsó eleme utánra!!!

STL

Standard Template Library
(a szabványos C++ sablonkönyvtár)

1) Tárolók
   std::vector < Int > vektor;
2) Iterátorok
   std::vector < Int >::iterator iter;
3) Algoritmusok

// tároló elejére stb.
iter = vektor.begin ();

// lehet növelgetni
++iter;
//
…*iter…

```
std::copy(v.begin(), v.end(), std::ostream_iterator<TudatMinta>(std::cout, "\n"));
std::sort(v.begin(), v.end(), tudatRendezes());
std::copy(v.begin(), v.end(), std::ostream_iterator<TudatMinta>(std::cout, "\n"));
```

Algoritmusok

```
typedef std::bitset<50> TudatMinta;

struct tudatRendezes
{
    bool operator()(const TudatMinta& a, const TudatMinta& b) {
        LZWBinFa afa, bfa;
        for (int i=0; i<a.size(), i<b.size(); ++i) {
            afa << a[i];
            bfa << b[i];
        }
        return bfa.getSzoras() < afa.getSzoras();
    }
};

int
main (int argc, char *argv[])
{
    std::vector< TudatMinta > v;

    TudatMinta elso(std::string("11111111111111111111111111111111111111111111111111"));
    TudatMinta masodik(std::string("11111000001111100000111110000011111000001111100000"));
    TudatMinta harmadik(std::string("01100101100111110010011011001100001100110010110111"));
    v.push_back(elso);
    v.push_back(masodik);
    v.push_back(harmadik);

    std::copy(v.begin(), v.end(), std::ostream_iterator<TudatMinta>(std::cout, "\n"));
    std::sort(v.begin(), v.end(),  tudatRendezes());
    std::copy(v.begin(), v.end(), std::ostream_iterator<TudatMinta>(std::cout, "\n"));
}
```

Kimeneti iterátor (a << operátorral ír ki)

Algoritmusok

```
class MentalFingerprint
{
    std::string owner;
    std::bitset<50> sample;
    double var;
public:
    MentalFingerprint(std::string owner, std::bitset<50> sample): owner(owner), sample(sample) {
        LZWBinFa f;
        for (int i=0; i<sample.size(); ++i)
            f << sample[i];
        var = f.getSzoras();
    }
    double getVar() const {
        return var;
    }
    std::string getOwner() const {
        return owner;
    }
    std::bitset<50> getSample() const {
        return sample;
    }
    bool operator<(const MentalFingerprint& mfp) {
        return var < mfp.getVar();
    }
    friend std::ostream& operator << (std::ostream & os , const MentalFingerprint& mfp)
    {
        os<<mfp.getOwner() << ":" <<mfp.getSample() << "[" << mfp.getVar() << "]";
        return os;
    }

};
```

Algoritmusok

```
struct tudatRendezes
{
    bool operator()(const MentalFingerprint& a, const MentalFingerprint& b) {
        return a.getVar() < b.getVar();
    }
};

int
main (int argc, char *argv[])
{
    std::vector<MentalFingerprint> v;

    std::bitset<50> jociMintaja(std::string("11111111111111111111111111111111111111111111111111"));
    std::bitset<50> muciMintaja(std::string("11111000001111100000111110000011111000001111100000"));
    std::bitset<50> cucuMintaja(std::string("01100101100111111001001101100110000110011001010110111"));
    std::bitset<50> duduMintaja(std::string("00000000000000000000000000000000000000000000000000"));
    std::bitset<50> lalaMintaja(std::string("00000000000000100000000000011100000000000010000000"));

    MentalFingerprint joci("joci", jociMintaja);
    MentalFingerprint muci("muci", muciMintaja);
    MentalFingerprint cucu("cucu", cucuMintaja);
    MentalFingerprint dudu("dudu", duduMintaja);
    MentalFingerprint lala("lala", lalaMintaja);

    v.push_back(joci);
    v.push_back(muci);
    v.push_back(cucu);
    v.push_back(dudu);
    v.push_back(lala);

    std::copy(v.begin(), v.end(), std::ostream_iterator<MentalFingerprint>(std::cout, "\n"));
    std::sort(v.begin(), v.end(), tudatRendezes());
    std::cout << "----------------------------------------------------" << "\n";
    std::copy(v.begin(), v.end(), std::ostream_iterator<MentalFingerprint>(std::cout, "\n"));

}
```

Algoritmusok

```
template <int size>
class MentalFingerprint
{
    std::string owner;
    std::bitset<size> sample;
    double var;
public:
    MentalFingerprint(std::string owner, std::bitset<size> sample): owner(owner), sample(sample) {
        LZWBinFa f;
        for (int i=0; i<sample.size(); ++i)
            f << sample[i];
        var = f.getSzoras();
    }
    double getVar() const {
        return var;
    }
    std::string getOwner() const {
        return owner;
    }
    std::bitset<size> getSample() const {
        return sample;
    }
    bool operator<(const MentalFingerprint& mfp) {
        return var < mfp.getVar();
    }
    friend std::ostream& operator << (std::ostream & os , const MentalFingerprint& mfp)
    {
        os<<mfp.getOwner() << ":" <<mfp.getSample() << "[" << mfp.getVar() << "]";
        return os;
    }

};
```

Algoritmusok

```
template <int size>
struct tudatRendezes
{
    bool operator()(const MentalFingerprint<size>& a, const MentalFingerprint<size>& b) {
        return a.getVar() < b.getVar();
    }
};

int
main (int argc, char *argv[])
{
    const int size = 50;

    std::vector<MentalFingerprint<size> > v;

    std::bitset<size> jociMintaja(std::string("11111111111111111111111111111111111111111111111111"));
    std::bitset<size> muciMintaja(std::string("11111000001111100000111110000011111000001111100000"));
    std::bitset<size> cucuMintaja(std::string("01100101100111110010011011001100001100110010110111"));
    std::bitset<size> duduMintaja(std::string("00000000000000000000000000000000000000000000000000"));
    std::bitset<size> lalaMintaja(std::string("00000000000001000000000000111000000000000010000000"));

    MentalFingerprint<size> joci("joci", jociMintaja);
    MentalFingerprint<size> muci("muci", muciMintaja);
    MentalFingerprint<size> cucu("cucu", cucuMintaja);
    MentalFingerprint<size> dudu("dudu", duduMintaja);
    MentalFingerprint<size> lala("lala", lalaMintaja);

    v.push_back(joci);
    v.push_back(muci);
    v.push_back(cucu);
    v.push_back(dudu);
    v.push_back(lala);

    std::copy(v.begin(), v.end(), std::ostream_iterator<MentalFingerprint<size> >(std::cout, "\n"));
    std::sort(v.begin(), v.end(), tudatRendezes<size>());
    std::cout << "----------------------------------------------------" << "\n";
    std::copy(v.begin(), v.end(), std::ostream_iterator<MentalFingerprint<size> >(std::cout, "\n"));

}
```

STL tárolók áttekintése

http://www.cplusplus.com/reference/stl/

Member map

This is a comparison chart with the different member functions present on each of the different containers:

			Sequence containers			Associative containers				
Headers			<vector>	<deque>	<list>	<set>				<bitset>
Members		complex	vector	deque	list	set	multiset	map	multimap	bitset
	constructor	*	constructor	constructor	constructor	constructor	constructor	constructor	constructor	constructor
	destructor	O(n)	destructor	destructor	destructor	destructor	destructor	destructor	destructor	
	operator=	O(n)	operator=	operator=	operator=	operator=	operator=	operator=	operator=	operators
iterators	begin	O(1)	begin	begin	begin	begin	begin	begin	begin	
	end	O(1)	end	end	end	end	end	end	end	
	rbegin	O(1)	rbegin	rbegin	rbegin	rbegin	rbegin	rbegin	rbegin	
	rend	O(1)	rend	rend	rend	rend	rend	rend	rend	
capacity	size	*	size	size	size	size	size	size	size	size
	max_size	*	max_size	max_size	max_size	max_size	max_size	max_size	max_size	
	empty	O(1)	empty	empty	empty	empty	empty	empty	empty	
	resize	O(n)	resize	resize	resize					
element access	front	O(1)	front	front	front					
	back	O(1)	back	back	back					
	operator[]	*	operator[]	operator[]				operator[]		operator[]
	at	O(1)	at	at						
modifiers	assign	O(n)	assign	assign	assign					
	insert	*	insert	insert	insert	insert	insert	insert	insert	
	erase	*	erase	erase	erase	erase	erase	erase	erase	
	swap	O(1)	swap	swap	swap	swap	swap	swap	swap	
	clear	O(n)	clear	clear	clear	clear	clear	clear	clear	
	push_front	O(1)		push_front	push_front					
	pop_front	O(1)		pop_front	pop_front					
	push_back	O(1)	push_back	push_back	push_back					
	pop_back	O(1)	pop_back	pop_back	pop_back					

Tervezési minták

tartalmazás

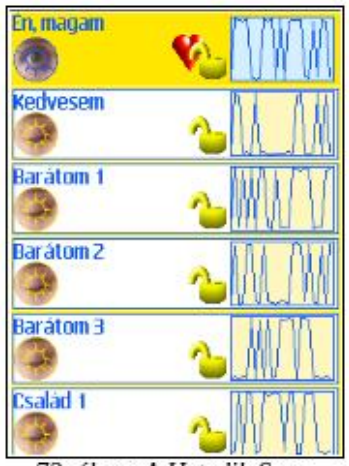
```
class BitMinta {
        public:
                BitMinta(string)…
}
class MentalFingerprint {
               BitMinta minta;
      public:
               MentalFingerprint(BitMinta minta):minta(minta) {}
}
```

```
class MentalFingerprint {
               BitMinta* minta;
       public:
               MentalFingerprint(string minta):minta(new BitMinta (minta)) {}
               ~MentalFingerprint(){delete minta;}
}
```

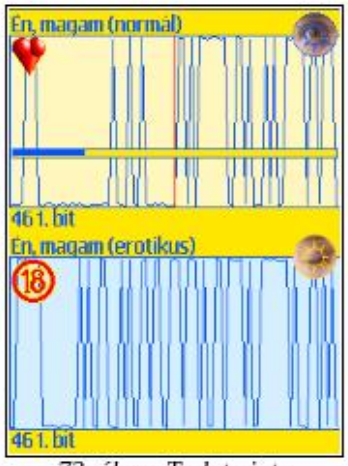
```
class MentalFingerprint {
               BitMinta& minta;
       public:
               MentalFingerprintstring minta):minta(* new BitMinta (minta)) {}
               ~MentalFingerprint(){delete &minta;}
}
```

Labor – Hetedik Szem

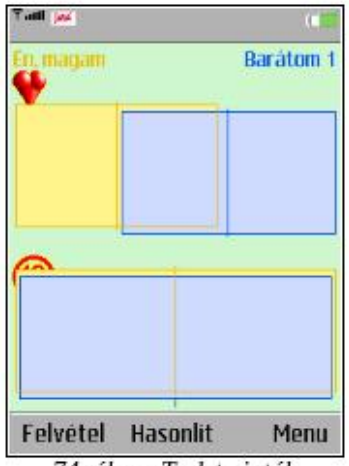
71. ábra: A Hetedik Szem indító képernyője.



72. ábra: A Hetedik Szem menüje, egyben a lenyomatok listázása..



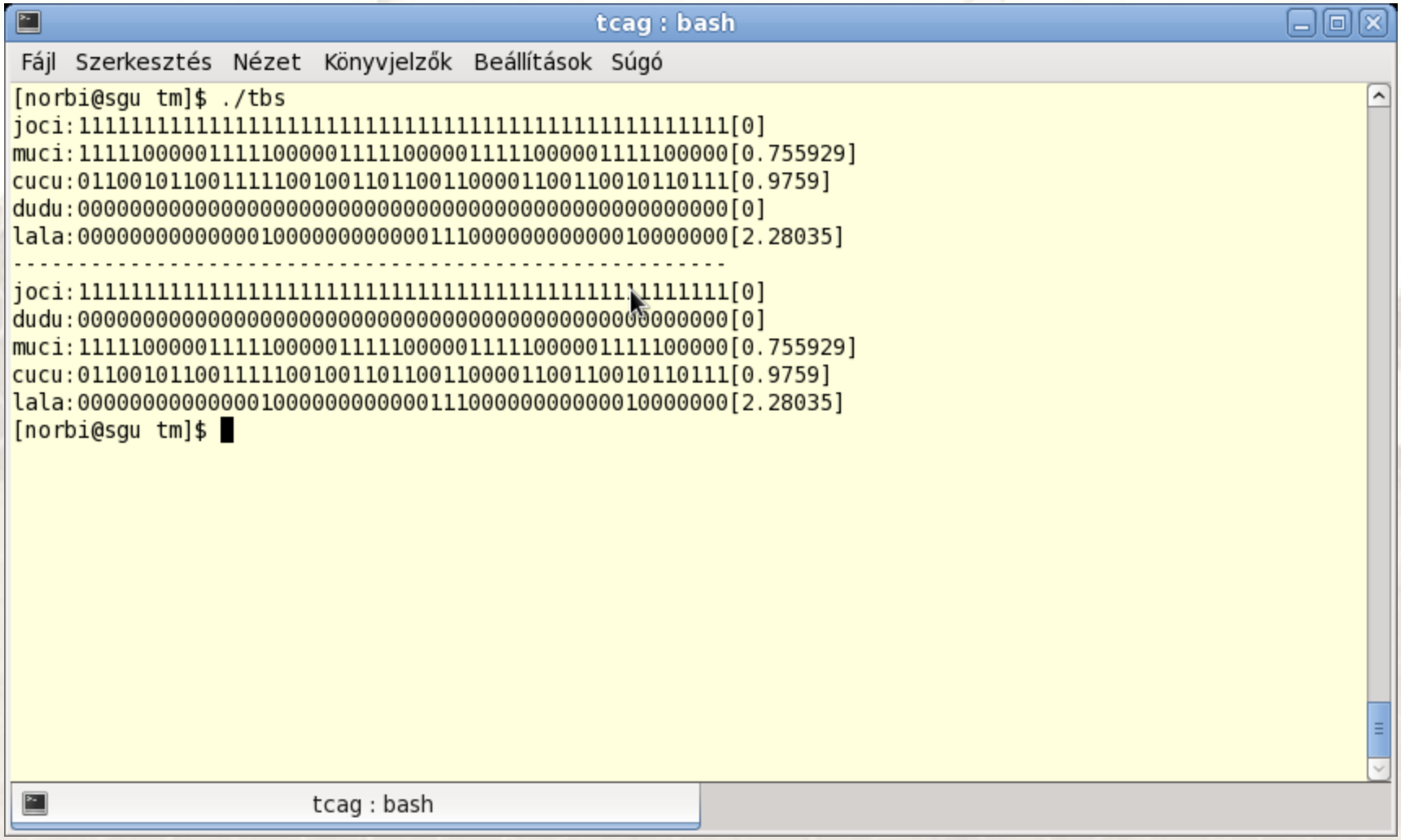
73. ábra: Tudatminta rögzítése (a „Szabad Akarat Szonda").



74. ábra: Tudatminták összehasonlítása.

http://progpater.blog.hu/2011/04/24/tudatmintak_rendezese

Labor – Hetedik Szem



http://progpater.blog.hu/2011/04/24/tudatmintak_rendezese

(Ismétlő) laborkártyák

```c
#include <stdio.h>

int fgv1(int i) {return 42;}
int fgv2(char c) {return 24;}
int fgv3(double d) {return 0;}

typedef int (*FGV)(int);

int
main ()
{
  FGV fgv;
  fgv = fgv1;
  double param = 5.5555;
  printf("%d\n", fgv(param));
  return 0;
}
```

Mit ír ki? (Ha lefordul… s miért?)

(Ismétlő) laborkártyák

```
#include <iostream>
#include <iomanip>
#include <bitset>

void
kiir(int e)
{
 std::cout <<
   std::setw (10) << std::dec << e <<
   std::setw (10) << std::hex << e <<
   std::setw (20) << std::bitset<16>(e) << std::endl;
}

int
main (void)
{

 int e = 0b0001000101010101;
 int m = 0b0000010000100100
 kiir(e);
 kiir(m);

 int a = e & (~m);
 kiir (a);

 a = e | m;
 kiir(a);

 a = e ^ m;
 kiir(a);
}
```

Magyarázd meg az 5 eredménysor bitjeit!

4437	1155	0001000101010101
1060	424	0000010000100100
4433	1151	0001000101010001
5493	1575	0001010101110101
5489	1571	0001010101110001

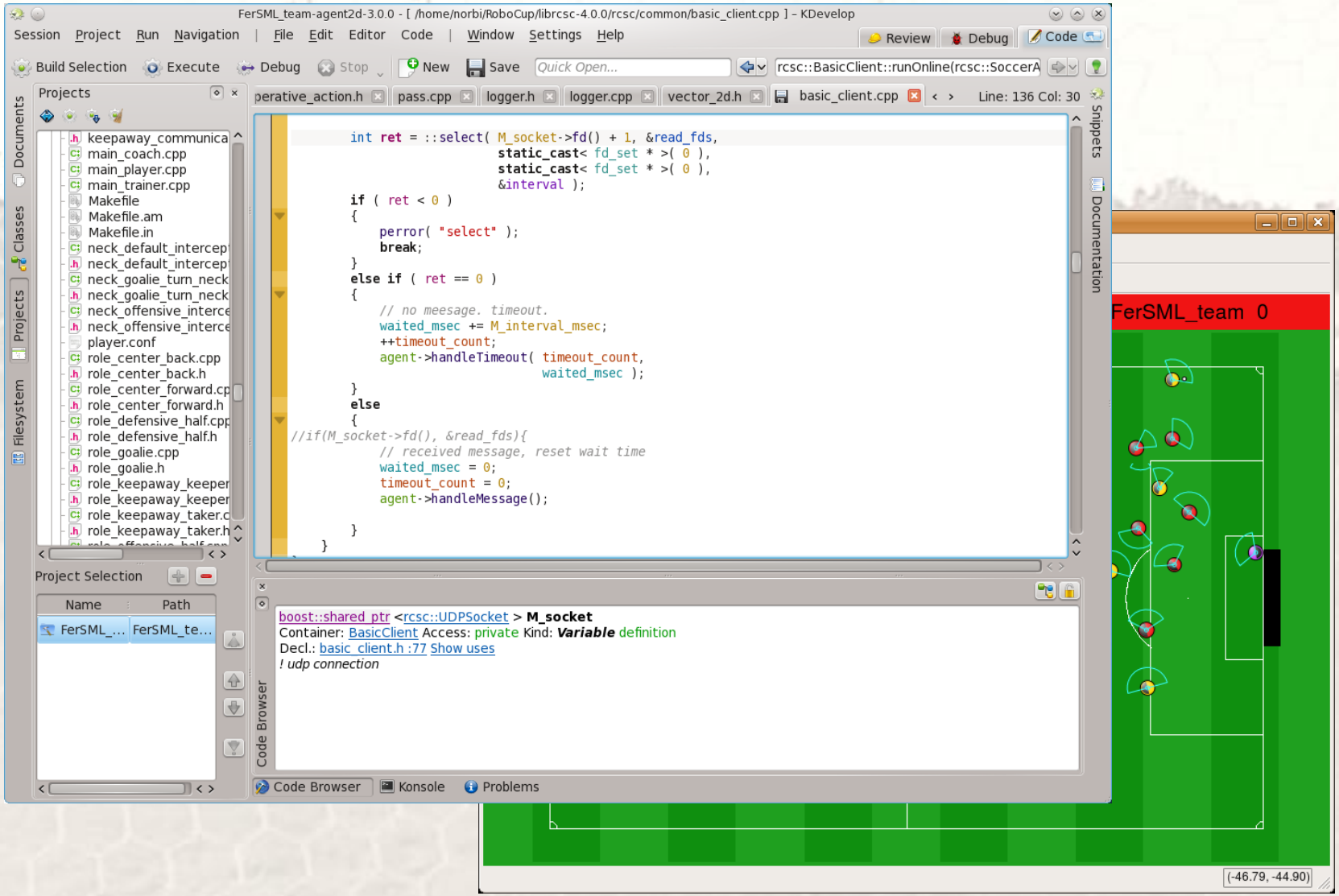
Laborkártyák

```
#include <iostream>
#include <vector>
#include "int.h"
int
main (int argc, char *argv[])
{
  std::vector < Int >vektor;

  vektor.reserve(4);

  Int ketto(1);

  std::cout << "vektor.push_back (1);" << std::endl << std::flush;
  vektor.push_back (1);
  std::cout << "vektor.push_back (ketto);" << std::endl << std::flush;
  vektor.push_back (ketto);
  std::cout << "vektor.push_back (41);" << std::endl << std::flush;
  vektor.push_back (41);
  std::cout << "vektor.push_back (42);" << std::endl << std::flush;
  vektor.push_back (42);

  for (std::vector < Int >::iterator iter = vektor.begin ();
       iter != vektor.end (); ++iter)
    std::cout << *iter << ", ";

  std::cout << std::endl;
  return 0;
}
```

Mikor-mi fut le, használd az Int osztályunkba épített „laborkártyás logolásunkat”!

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.

264-296

(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

431-452

„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

214-219

(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

465-480

„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