

döödas

7/11

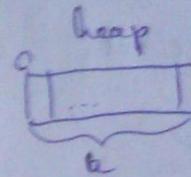
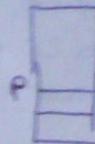
Előtérben szabolyai

- automatikus változók
- lokális statikusok
- globális / hozter / osztály statikus tag
- dinamikus objektumok → heap-en alkotott objektusok

int k;

int *p = new int(k);

stack



delete [] p;

int *q = new int(k);



delete q;

ha elvezetem a pontert, nem tudom kitörölni
a tét => memória "szivorges" lep fel,

ha kitörölök a programot, akkor felszabadítja

matrix

int **m = new int*[k];

- 2x meghívott delete → nem definált
hiba, összefonhatatlan memória
leapunk

→ rossz delete → fates időtű hiba

for (int i=0; i<k; ++i)
{ m[i] = new int[k]; }

for (int i=0; i<k; ++i)

{ delete [] m[i]; }

delete [] m;

→ tömb törlés

- Tömb elérés

Tömbelérés:

- Osztály adattag

- Temporális objektumok

Tombelen: letrejön tömb letírásakor
megszűnik: tömb megszűnik

7/12

Objektus adattag: letrejön tömbben objektum letírás
megszűnik: a tömbben objektum megszűnik

Temporális objektumok

$$x = \underbrace{(a+b)}_{\text{Temp obj}} + \underbrace{c}_{\text{Temp obj}} + \underbrace{d}_{\text{Temp obj}}$$

eztől az objektumokat hozza leírja,
ha végzett a (+)-művelet előtt a T_2 -et
kennyezje az x helyére majd megszűnik.

Létrejökhöz: a részhelyezés kiértékelésével

Megszűnések: minden művelethez a fejlesztő

std::string a,b;

const char* p = (a+b).c_str();
if (strlen(p) < 20)
{ ...
3
} ...
→ leírta a teljes műveletet, hogy mi legyen leírva a C-string
es (a+b) megszűnés
⇒ nem definiált, hogy mi legyen leírva a
p destruktője, mert nem fog
futtatni, és ekkor nem tudja ki-
hova fog mutatni a pointer.

char* answer(char* q)

char* answer (char*q)

{ std::cout << q;
char ans[80];
std::cin >> ans;
return ans;

↳ mi van /o/ ig a stringet?
char* → C-string /o/ ig mi /o/

std::cout << "Hello World" << endl

Problémák: valódi mutató pontírozás

↳ kihívá: pontírozás adott műveletekhez, valódi, keleti
eljárás: forrásból adatfelvétel felelet felülírása
80 char -rel többet írhat, mint a max

std::cin.getline(ans, 80); → ha annál többet ír-e ki a 27

(igazolás) ↳ (figyelmeztetés)

char ans[80] //igazolás valtozó

char* answer (char*q)

std::cout << ans << ("WTF");

jól fog működni

std::cout << answer("Hogyan vagy?") << answer("Bízd!");

7/2

std::cout & std::endl ("Hello World")

std::cin & std::cout ("Hello World") << endl ("Hello World")

→ kein freigabe nötig, da C++ Wörter sind

→ in 2. Zeile hat C++ eigentlich nur einen freien N.

char & cin << str;

char & cout << str;

1

2

→ in Wörtern liegt ein Wertatlas vor. 2x wiederverwendung von j.

char *p = cin << str; // standard ist der Wertatlas, wenn die beiden
delete[] p; doppelten Angaben nicht mehr nötig für eine Atla

std::string ausdruck (char *p)

{ std::cout << p;

std::string aus;

std::getline (std::cin, aus);

return aus;

}

string: automatische Konstruktion und Zerstörung

mit dem Spezialoperator

Parameterübersetzung

- mit -oer: mit parameterübersetzung

- C++ -parametri

- erzeugen Parameter

- mit -P / -param - param

- neu - variab

C++ - Standard

void f(int x)

{ formal parameter

f(x)

→ Aktivierungsparameter

Wertübertragung

zu Zu-Parameter
durchgängig möglich

→

7/3

cout < i; → cout < i
Start (%d, &x);

7/74

C++ variablen parameter verändern

C++ kon: referenzen - alther

cout < a = 5; a = 5

cout &a = a;

c = f();

void ^{inc} f(^{int} &ⁱx)

{
 ⁱx++;
}

std::cin >> x;
 ^{inc}
 f(x);

→ takes built reference!!

void g(^{const} int&x)

{
 ³
}

object class k.

cout << x + 1 << endl;

de x++ new next Target!

++

da str C = "Hello" ;

std::cout << str[1] << std::endl; // a

std::cout << (str + 1) << endl; // b letter ello

std::cout << (str + 2) << endl; // gg

std::cout << (str[3] = 'e') << endl; // e

cout << x; → ambe pds
Scarf ("%" d, &x);

₹14.1
ea

Cum - szervíti parameter átadás

C++ban: referencia - átadás:

cout a = 5; arr
5

cout &a = a;

r = f;

void ^{inc} f(^{inc} cout &x)

{
 ^{inc} x;
}

std::cin >> x;
^{inc} f(x);

→ nincs hűl referencia!!

void g(const int&x)

{
 ...
}

Object C-szerele k.

cout << x+1 << endl;

de x++ nem! net temporary!

++

char str C J = "Hello"; i

std::cout << str[1] << std::endl / a'

ffjate
+ -

₹ 4.2
ea

std::cout << (str + 1) << std::endl // kibocsátja Hello

std::cout << (str[2] + 1) << endl // g8

std::cout << (str[2] = 'e') << endl // e

7/2
 #include <vector>
 #include <algorithm>
 #include <iostream>
 #include <iterator>
 int main()
 {
 std::multiset<int> m;
 std::istream_iterator<int> (std::cin),
 std::istream_iterator<int>();
 std::for_each(m.begin(), m.end(), std::copy(m.begin(), m.end(), std::ostream_iterator<int>(std::cout, " ")));
 }

Yannick
 7/2
 Soc
 8/1

8.cpp elbaada's

8.cpp gyasrat

8/1

[rc] cm
 ~~~~~  
 complex

double arc, amic,  
 bre, bim;

[http://ugod/pmo2/html/06\\_date/datemain.cpp.html](http://ugod/pmo2/html/06_date/datemain.cpp.html)

(date G4tterse was a double  
hoeven meer dertig?)

date x(2012, 11, 12);

! date & → // van hoge kwaliteit

| PL Java-ban een horraa weg

date d1(2004, 3); → date d1↑  
 [dit levert laag]  
 left, cert, ha mar en teelot  
 alleen nog

date s("2012-66"); →  
 Ott-ban ee een opg van

dit is leper loggen lezen;

while (true)

{  
 cout << (dt = 40) << endl; → er mis?

cout << dt <<  
 d2 ;  
 s ;

CPP  
SD  $n + c = \text{X}[\text{get\_year}(),$   
names we've added earlier]

date d(C),  
mais par contre ce  
sont des bases de la base de l'objet  
ce sont pour certains des bases  
lancées du déclaratif  
et donc elles sont dans le CPP.

const date birthday (1982, 2, 26);

stol::cout << birthday.get\_day();

class & operator ++() { return next(); }

date operator++(int){~~date~~

{ date curr(\*this), next();

return curr[3];

→ neu belegt  
(fehlterkeln)

(Wolterkens)  
Visse teken  
(1)

Epomophorus

Ms. 9.001.2 v. 4

meilleur

## Prefixes

1

OBRA

operator +

operator + (int) sum = 20

operator  $\oplus = (\text{int } n) \rightarrow \text{return odd}(n)$

obj were often seen

(a)  $f \rightarrow$  retransl.

Region Ural'skiy  
and eastern Kazakhstan

$\dots f(a==b)$   $\rightarrow$  a. operator  $== (b) \rightarrow$  tag fu,  $\hookrightarrow$  method?

8/3  
CPP  
La

3

(†(α = 2013)

23

Reflected, we

if (a < 2013)

乞  
3

©·pparter (( 2013 ))

↳ konversiðs operatorkel  
wegluvoddik, lete fyrir es  
date(2013) 2013.01.01

if ( $2013 > a$ )

三

2013. operator > ( $\alpha$ )

Off 9/30

8/5

C++ Ea

9/11

9. old class

date d(2012, 11, 19);

operator <<

operator ?

2 operator types you

std:: cout << ;

operator << (std:: cout, cl);

std:: cout << operator << (d);  
ostream

ostream & operator << (const date& d);  
↓

friend

std:: istream & operator >> (std:: istream & is, date& d);  
istream & operator >> (std:: istream & is, const date& d);

inline friend operator

date a(2012)

date b(2012, 12, 6)

if (a < b)

operator <(b)

operator <(a, b)

d.setyear();

this → kein eigener return  
punktet, modifiziert

this teilt

setyear(&d, 2013)

this

date\*

const → keine autoritätsre. neg

cl. getMonth()

getMonth(&d)

const date\* this

bool operator <( date d1, date d2 )  
while bool operator == ( ... ) { }

if(2013>6)

else one also has global variables

2013 operator >(b) → b[3]

operator >(2013b)

2 elements enter  
at almost

b[0] van enkelte a 2013, aktor ða fyrir ða man er ferzgjörður  
at, og b[3] bærar bær ar ekki.

Arfendur meðileg orðið að þá privat fyrir heildarhl.  
atari??

if (y>m>d)

{ self(y,m,d);

}

jobbinniða rígtileg að valtorkun  
slávinnið, heft í og hefur  
3 daga, vega tilbúar aktor  
nein býja fellur,

Ret : void date :: print(stat::streng const)

os<<"["< get\_year()< ", \*< get\_month...;

↳ he valtorki að fáði ress, eð. atdil myg  
hefðist allt hér.

→ bool opa <

mett a privat d2 d1 - hefur ekki fátt að ekki er

getmonth() → nem závarð að fá hér, man verður  
sótt, hvern a fréttarinni völum hérðar að hér!

Pt 2/2  
Myndar að fyrir heim  
leifar be  
leikur 2013 operator

if(b<2013)

else dateinniði  
alarathnir

til fyrir skrifstofur  
(man street for sign up)

Att eel

9/3

assimilatice.h / ~good\_prg2 / hfile / ex7-pool /  
(ex6-pool)

declaration, compilation

dataclass d1(10);

friend operator: (global friend)

nem lajpa a privat member as externez

→ public interface nemmel jo ujto ujlelhet

ezelst ~~3~~ operatorral meghozes

mp

(s-anything) ? 0, 0 : v[->sp]

↓

variable address

new address

elisse a 2-sp, aleszt?

elisse konziderit

azon arra hivja

meg a v3-nt.

destructor:

- nincse parameter

↳ nem lehet

tiltathetni

Copy:

(const dstack & other)

szabaly: referenciait kell megadni; nem  
nem műsor, mindenbe a copy + minden megvan  
azon valószinuval refektőrt elrejtve.

(v[->sp])

~~v[->sp] = ]~~

~~new~~

~~operator~~

~~left hand~~

~~operator~~

dataclass d1(10)

d=d1x

d=fi

dataclass

Mi eszel abaj?

→ eszel hasznalatkozog felirat

referenciat

copy(other)

ha az eszerinti eszerinti  
elvezet az eszerinti  
lehet ugyan

lehet ugyan

# Sablonok, template -02

tipusai törlő parameteresets

10/  
80

#define MAX(a,b) ((a)<(b)?(b):(a)) → minden tipus ilyen helyen működik, de hivatalosan

malloc, free void\* pointer típus

void\* malloc (int bytes);  
void free (void\*);

OppEa  
10/11.1

cout printf (char\* str, ...);

template< classT >

void swap (T& a, T& b)

{ T tmp = a;

    a = b;

    b = tmp;

}

Függelék  
- önmagában nem használható  
→ példányosítani kell

int x = 6;  
int y = 4;  
swap(x,y); → hosszba!;  
[T = int]

T tmp = a;  
a = b;  
b = tmp;

}

C++ 03:

- funkciók (módszerek)
- típus sablonok

int x = 6;  
int y = 4;  
swap(x,y); → hosszba!;  
[T = int]

- bekezethető → esetrajtja  
General a forrólókog swap funkciót

General a forrólókog swap funkciót

egy swap funkciót

példányosítás

template< classT >

class List

{

};

→ List<int> a;

OppEa  
10.11.2

Nem mindenek nevező semmilyen példig elérhetőké vannak pl.  
itt hogy tmp = a; → copy kontraktora legyen, ex-fordítás operatora  
legyen  
ha le van tiltva a copy ⇒ baj lesz!, fordítási hiba!  
fordító program eszerrexei.

## Specializációk

- részleges specializáció
- teljes specializáció

template <ClassT>

class Matrix

{

}

template <ClassT>

class Matrix<T\*>

{

};

Cpp Ea  
10/2.1

template <ClassT>

class Matrix<T\*>

{

...

};

template () teljes

class Matrix<~~open~~<sup>bool</sup>>

{

};

void \* malloc (int bytes);

void free (void \*);

Set : random forrásba os adathalm

template <ClassT, class Comp = std::less<T>>

class set;

std::set<int> a;

std::set<int, std::greater<int>> b;

Template parameterek:

- típus

- integrális konstansok: int, bool

- (különböző összetételek) funkció mutatók ponterek

tag funkció objektumra mut. po-

~~template <ClassT>~~

~~class Vector~~

{ T \* pi;

public

Vector (const

Faktoriyalıst szamolni forditási időben

10/3  
ea

template <int N>

struct Factorial

{

enum Value = Factorial<N-1>;

Factorial<N-1>::Value Zi;

Zi

CppEa  
10/3.1

template <>

struct Factorial<0>

{ enum Value = 1; Zi;

Zi

std::cout << Factorial<5>::Value;

120

- forditási időben számol
- faktársi időben már csak leír.

template <class T, int N>

class Array

forditási időben időut

{ T v[N];

};

vars:

array <int 10> a;

int k;

std::cin >> k;

//Array <int 4> b;

↳ forditási hiba

array <int 12> c = a; törzsenek kell a művelet

class Vector

{ T \* p;

public:

Vector (int n)

{ p = new T [n];

};

hiba: nem egyszer meg a Tipusai => hiba!

Vector<int> v(10);

Vector<int> w(12);

v = w;

w = v;

# Standard Template Library (STL)

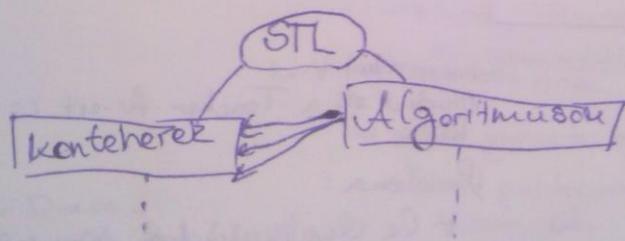
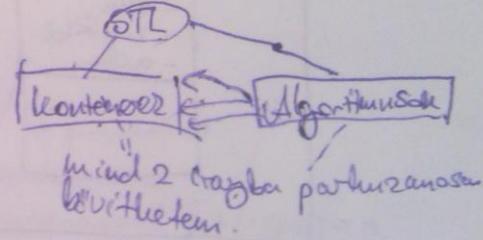
Generikus programozás

OPPEA  
11.1.1

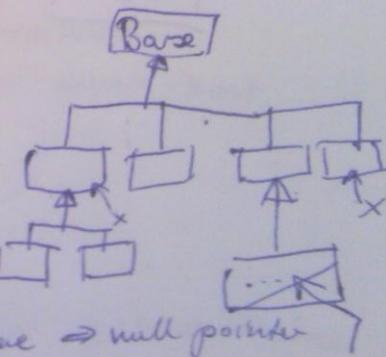
- Adatstruktúrak (konténerek)

std::list, std::vector

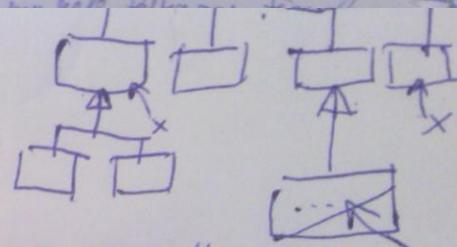
- Algoritmusok - konténer független



elemeikben Class-al  
fizzetve, ott eamel Class-ral  
ellenőrizni tipushalásba  
lennek.



elemeikben Class-al  
fizzetve, ott eamel Class-ral  
ellenőrizni tipushalásba  
lennek.



Keresünk meg egy elemet a tömbben, ha nincs base  $\Rightarrow$  null pointer  
ha beire van mutátorral eger pointerrel

int\* find(int\* first, int\* last, int x)

```
{
    while (first != last)
        if (*first == x)
            return first;
        first++;
}
```

```
return 0;
}
```

}

OPPEA  
11.1.2

template <typename T>

T\* find(T\* first, T\* last, const T&t)

{ while (first != last)

{ if (\*first == t)

{ return first; }

first++;

}

return 0;

$\rightarrow$  tömbökön iterálva, hogyan  
tudunk ottől elkezdeni?

template <typename T, typename Iterator>

Iterator find (Iterator first, Iterator last, const T&t)

```
{  
    while (first != last)  
    { if (*first == t)  
        { return first;  
        }  
        ++first;  
    }  
    return last;  
}
```

OppEa  
11/2.1

std::vector<int> v;

std::vector<int>::iterator i = std::find(v.begin(), v.end(), 5);

template <class Iterator, class Pred>

std::vector<int> v;

std::vector<int>::iterator i = std::find(v.begin(), v.end(), 5);

template <class Iterator, class Pred>

Iterator find\_if (Iterator first, Iterator last, Pred p)

```
{  
    while (first != last)  
    { if (p(*first))  
        { return first;  
        }  
        ++first;  
    }  
    return last;  
}
```

OppEa  
11/2.2

# M13 Iterator

Struktur X

{

bool operator () (const x) const

{ return x < s;

}

};

11.3.1  
Cop Ea,

Zh: STL es adaptierter  
Standard-Klassendesign.

std::list<int> c;

std::list<int>::iterator it =

std::find\_if(c.begin(), c.end(), X());

Adapt sequenziell:

// http://www.digi.com/tech/stl

- Sequenziell's Konstrukte:

std::list

// http://www.digi.com/tech/stl

- Sequenziell's Konstrukte:

std::list

std::vector

std::deque

11.3.2  
Cop Ea,

- Assoziativ adaptierter:

- random acc.: set, multiset, map, multimap

- random acc. unordnet

- Adapter:

stack, queue, priority-queue

(Template param: sequentielle Konten)

## Iterator hierarchie

std::list - Bidirectional

std::vector, std::deque - Random Access

("regulars")

For Word Iterator  
Output Iterator

template < class Iterator >

void f(Iterator first, Iterator last)

{  
    type name | std::iterator\_traits<Iterator> value\_type a = \*first;

...  
}

OppEa  
11.4.1



int x;

template < class T >

Foo <T>: N \* x;

}

erőszakos  
painter  
Népszerű X emi.

↳ erőszakos szavazás területi

ha azt szeretnék meg típus  
typename N



std::list<int>::const\_iterator

}

ha azt szeretnék meg típus  
typename N

std::list<int>::const\_iterator

OppEa  
11.4.2

std::set<int>::reverse\_iterator

std::deque<double>::const\_reverse\_iterator

map: asszociatív tömb

std::map<std::string, int> S;

S["blabla"] = 5;

↳ hazzal ezz vagy megbízhat a művelet, mert nem hivatalos elérési

s. find("abc")