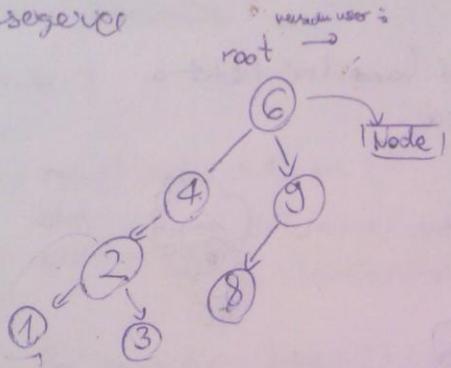


7. gyakorlat

7.1

F: számolásit olvassuk be (eof-ig) rendszerrel kijjss, Beharr's körülött  
segít-szervicer



Cpp Gyak  
7/1.1

in order bejárás: rendszerrel fogja  
leírni

struct Node  
{ int val;  
Node\* left, \*right;

Node (int v)  
{ val = v;  
left = right = 0;

Cpp  
Gyak  
7.1

void print (const Node\*n)  
{ if (n) // nem 0 pointer  
{ print (n->left);  
std::cout << n->val << std::endl;  
print (n->right);  
}

3

n->left  
~~~~~  
(\*n).left  
~~~~~  
dereq.

Cpp Gyak  
7/1.2

head->next->next->next->value

C-ban  
Node \*n  
if (\*n)  
... ~  
\*n = new Node (v);

↑  
Node \*n  
↓  
akkor a  
root pontszám  
meg.

void insert (Node\*&n, int v)  
{ if (n)  
{ insert (v <= n->val ? n->left : n->right, v);  
}  
else  
{ n = new Node (v);  
}

3

a baj:  
root 0 pointer maradt

bal részt jobb részt → delete!

6 8 12 ≠ 3  
csof

> /előtérük

int main()  
{ int i;  
Node \*root = 0;  
while (std::cin >> i) insert (root, i);  
print (root);  
// ...  
}

}

## Standard Template Library

#12

C++ darüber hinaus

<http://www.sgi.com/tech/stl>

STL: Kästen / algorithmen  
 (vector) (for each)

→ parameterisiert; kann leicht überladen

C++ Eh - n führt (ausserdem) mehrere  
 Objekte ein



Start Typ  
 C++ Obj. orientierung

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

cut main()

```
{ int i;
std::vector<int> v;
while (std::cin >> i)
    v.push_back(i); // vector wächst um i erweitert
std::sort(v.begin(), v.end());
```

```
for (int i = 0; i < v.size(); ++i)
    std::cout << v[i] << " ",
```

// std::for\_each (v.begin(), v.end(), print)

3

- fiktivs Objekt Wörterbuch ↑
- für: Klassische Schleifen

newer indexieren

```
#include <set>
#include <algorithm>
#include <iostream>
void print (const int& i)
{ std::cout << i << "
```

cut main()

```
{ int i;
std::multiset<int> m;
while (std::cin >> i)
    m.insert(i);
std::for_each (m.begin(), m.end(), print); }
```

3

④ std::for\_each (m.begin(), m.end(),  
 print);

letztens zurück funktioniert diese Gedanken?  
 std::copy(m.begin(), m.end(), std::ostream\_iterator<int>(std::cout, " "));  
 std::copy(m.begin(), m.end(), std::ostream\_iterator<int>(std::cout, "\n"));

⑤ ist nicht möglich

→ std::multiset<int> m( std::istream\_iterator<int>(std::cin),  
 std::istream\_iterator<int>());

→ kann nicht

7/2  
 8/1  
 8/1  
 8/1

```


#include <vector>
#include <algorithm>
#include <iostream>
#include <iterator>

int main()
{
    std::multiset<int> m(
        std::istream_iterator<int>(std::cin),
        std::istream_iterator<int>());
}

m.for_each(m.begin(), m.end(),
           std::copy(m.begin(), m.end(), std::ostream_iterator<int>(std::cout, "")));


```

## 8.cpp eldada's

## 8.cpp gyaslat

8/1

rc | cm  
 ~~~~~  
 complex

double arr, arr;  
 bre, bim;

[http://ugod/pw2/html/06\\_date/datemain.cpp.html](http://ugod/pw2/html/06_date/datemain.cpp.html)

(date G4tterse was a double  
hoeven niet dertig)

date x(2012, 11, 12);

! date & → // van hoge kwaliteit

| PL Java-ban een horra nieg

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

date s("2012-66"); →  
 Ott-ban ee neen nieg van

dit is leper loggen lezen;

while (true)

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

cout << dt <<  
 d2 ;  
 s ;

12  
std::cout < answer ("UTF8");

std::cout < answer ("Hogyan vágy") << answer ("Bimbó");

→ nem tudjuk megbízni, hogy melyik meg lesz,

→ a 2. 2x fájl le, ugyanaz a változó többel ki.

char\* answer(char\*q)

{  
 std::char ans[256]; → leugyelő felület a fü körül

}

→ 1x hivatkozás a leírásból és 2x ugyanez a változó jön.

char \*p = answer ("delel?"); // szabadítja fel a töreget, mert bőlönbőr  
delelrogyni fog, és többször írásba kerül az előd  
delete[] p;

std::string answer (char\*q)

{  
 std::cout << q;  
 std::string ans;  
 std::getline (std::cin, ans);  
 return ans;  
}

string: műszer konstruktorral hívható meg

→ enter minden addig el

(CPP)

std::string answer (char\*q)

{  
 std::cout << q;  
 std::string ans;  
 std::getline (std::cin, ans);  
 return ans;  
}

string: műszer konstruktorral hívható meg

→ enter minden addig el

### Paraméter átadás

- akt - szerinti paraméter átadás
- cím - szerinti
- eredmény szerinti
- érték / cím - szerinti
- hív - szerinti

(CPP)  
7/3/2  
ca

### Erdék - szerinti

void f(int x)

{  
 ...  
}

f(4);  
 ↗ aktuális paraméter  
 ↗ ezt lokális  
 változóján írja

információk  
algoritmus irányában



## 8. gyakorlat / 8/1/

8/1/ Jeladat

std::set -ről es f -ig egy szöveget beolvassuk  
betűszámtípusokat:

a: 24

b: 36

c: 8

d: 3

e: 7

mu: 3

lf: sziszalekasz

start\_upper\_bound('a')

[aa ab bc cdde] elements

start\_lower\_bound('a')

a-val elválasztva előző karakter

```
#include <set>
#include <iostream>
#include <iterator>
```

int main()

```
{ std::multiset<char> stat;
  std::istream_iterator<char> (std::cin),
  std::istream_iterator<char>());
```

```
for( std::multiset<char>::iterator i = stat.begin();
```

```
i != stat.end(); i) // 3. rész üres
```

```
  std::cout << *i;
```

```
  << stat.count(*i);
```

i = stat.upper\_bound('\*');

// ha a \*i-vel egyező  
van ebben az iterátor  
adott karakterben.

azutolsó a-utca  
pontja

itratout al  
vissza



## Associatortomb



TOP SECRET  
8/2

```
#include <iostream>
#include <map>
#include<algorithm>
```

```
int main()
```

```
{
```

```
    char ch;
```

```
    std::map<char, int> stat;
```

```
    while (std::cin >> ch){
```

```
        stat[ch]++;
```

```
        ++stat[ch];
```

```
    }
```

```
    std::for_each (stat.begin(),
                   stat.end(),
                   print);
```

```
}
```

```
void print (const std::pair<char, int>& p)
```

```
{
```

```
    std::cout << c. first << " : "
```

```
        << c. second << std::endl;
```

void print (const std::pair<char, int>& p)

ha nem volt ideig eleny  
letrakerz eset ures bejegyzest  
ch-hoz

Letke 0-as bejegyzessel ekkor  
ann mi rogtan 1-re implementalhat  
-hosszat vers vagy nivaloget adhat!

Stabil-jel 1-el hozel a  
ch-shoz a reszet

Ha ungas ideig letrakerz  
egyet es rogtan megkulli. Oda  
1-re.

Összefoglalás

class X

{

3:

↳ *leendes műve*  
 $\text{sizeof}(x);$  ?

 $\text{sizeof}(x) = \text{sizeof}(\text{char});$ 

↓

egyszer (annel kevered nem)

Helyettesítés

X a;

X b(a);

X c=a,

a=b; // direkt ad operator

X \*p = &amp;a; // kompakt

// default constructor

// copy ctor

+ van destructor()

C++ban ~ Class class class public  
stun public

```
struct Foo
{
    Foo();
}
```

Parameter nélkül meghibásztatott konstruktör

Kein Wert + ~~standard~~  
default konstruktor

hivatalos a def. Ctor, ha:

- csak egyet

- semmilyen konstruktőr nincs

Foo a[5]; (5x hossza néz)

std::vector&lt;Foo&gt; v(5);

vector v[5] valtozó

Sorba Foo

konstruktör  
elérni

(1x hossza néz a konstr.)

→ a többi részhez

az osztály vector a másik rész

de mindez az 5 elemötökezetet hoz

Létre hozva az újra Foo -n

struct Foo

```
{ Foo(int i=16)
    { ... }
}
```

3

→ default  
param nélkül  
is meghibásztatott

Parametrikus  
nélkül cs  
meghibásztatott  
Foo() { }

val gyűjtött nem  
hibásított ⇒  
2x def.

copy ctor / entkoppelner operator

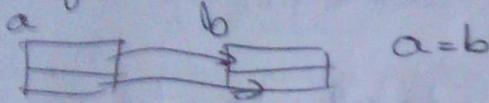
- plaudig Reihenfolge:

tagsammler maedas

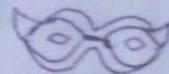
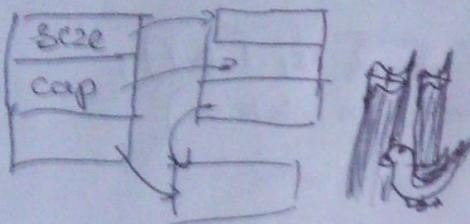
adattagant reihenfolge copy ctor ill.  
entkoppelner operator her.



Copy by



$$a = b$$



Foo(Foo rhs)

(right hand side)

↳ Foo x;

new! met

Foo y = x;

rhs

copy const-rhs

copy const

vegetable megjelenik a copy const-ot,

Foo{const Foo&} → always new megjelölhető!  
{ ... }

Foo & operator = ( const Foo& rhs )

$$x = y = z = a$$

referenciket  
visza adni  
hogy az megvalósított  
legyen!

9. operatorOperator ordring:

tag fu-selbst globaler Operator  
 - filterhelhetto: operator = operator == , operator []

- new filterhelhetto: ::, ., access, ?:

word op filterhelhetto, die new ogenbit

Operator och filterhelhetto:

Operator och, enligt globaler's minneleter

fix aritos: leveret: operator ()

(+, - > operator för till tuffare, )  
 van negativt att leter  
 + har 2 param-e van nu  
 leter ~~hab~~

## Class Vector

{ cut cap, si

T\* pi

T\* pi  
Public:

Vector (cut c) { cap c, s(0) }

} private

initialisering (3 ter  
 i adat begär mat fönster  
 källtypen är doc, brödell.  
 Ordet.

P = new T[c]i

}

Höga leter resul hazzabu:

Class  
 & int i,j;

Public

X (cut a) : j(a), i(j)

{}

}

eller a särskilda beteckn (4 ter)

förståelsehet j(a), i(j) X a(3)

a(i)

i(j), j(a) => new definierat  
 värde  
 med definierat värde.

## ~Vector()

{ delete [] p;

}

++  
jpe  
9/2

CH 11

1

Előző & operator = (const vector & this) = delete;

Vector (const Vector & this) = delete;

// Megadja a vector méretét

vector:  
Próbálkozás (this  
Delete művelet)

at Size () const { return s; } [this->s]

↳ ne olcsón megválasztani! ↳ this típusa const Vector \*

ne töltse

ez egy const-on a műveletek (const member)

Vector v (5), ↳ ez az objektnek a tagjai

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

// Indexelés operátorral adható el a elemet

// const vector vezetési eljárás

// de olcsón felülírni bár min const

cout v[3] = {3, 1, 6};

v[1] = 8;

v[0]++;

const cout c1[3] = {8, 4, 9};

std::cout << c1[1];

→ (tt ne olcsón írni, csak olvasni)

Tx operator[] (int idx)

{ return p[idx]; }

\*\*\*

const T& operator[] (int idx) const

{ return p[idx]; }

vector this

operator a  
T - vel, amely  
az adott változókat

(s)

ezek nem formál bele

1 fü - be 2dik kiell hozzá aminek  
nincs hozzájárulása

v[0]=6

T& → alakul.

+ v[1];

ez const - on működik  
ha const működik → ez ha nem az alk a  
feladat, ha nincs más, akkor ez fog  
lehasználni, ez azt jelenti hogy a műveletet CS

Class Matrix

```
{
public
    +& operator() (int x, int y)
{
    :   *
}
```

matrix m;  
 $m[1,3] = 5;$   
 $m[1][3] = 5;$   
 $m[1,3] = 5;$

$m[1,3] = 5$   
 $m[2,3];$

ezt árthatjuk, ez a legkönnyebb  
megérhető, a minden 2-köt  
tükörzni kell.

Vector operator+(const Vector& a, const Vector& b)

```
{
}
```

) elter személyi paraméter → copy konstr (am két tömb  
cpu + globalis 2 db. paramétere van.)

② ~~operator()~~ operator+(const, const)

```
void print (const int& i)
{
    std::cout << i;
}
```

ezt a paraméter parancsba hozó szerelhető fájlnak

tipus

class Point

```
{
    std::ostream& os;
public
    Print (std::ostream& o) : os(o) {}
```

void operator() (const int& i)

```
{
    os << i << ' ';
}
```

}

~~std::for\_each(v.begin(), v.end(), print(std::cout))~~

Print p (std::cout);

p(1);  
p(2);  
----

std::for\_each(v.begin(),

v.end(),

p);

kerülne az elnevezés



- leid ejra felhozatalas
  - polimorfizmus
  - (altipus - keprés)

Cap City  
10.111

```

std::vector<int> v(3);           (3 elem vector)
// std::string s(3); → Hiba      vector<string>
// std::string str(0); → ok       konstruktor, hashtab,
                               fordel
                               Szával működik?
                                → O pointerekhez! ezért fordelik (e.)
                                ⇒ őszintű

```

`std::string a("alma");`

- 1 cleme mutato' pointer var
- O → as os null pointer
- byg neme as adatath, de  
nem tud /O-ig nemni', mett  
nem mutat seholva.

Feladat:  $O \rightarrow$  letre jíjján az árás string, string konstruktorral!

class safe-strong  $\Rightarrow$  <sup>AppGlob</sup> classall

`std::string a("elma");` - 1 elme mutató pointer van  
- 0 → az a os null pointer  
Vagyis minden os adattípus, de nem tud 0 - ig lenni, mert nem mutat sehol.

Feladat: nem mutat behívva.  
0 → feltrejzián az üres string, string konstruktorral

```
class safe_string  
{  
    std::string str;  
public:
```

EppGyar  
10/1.2

ha P null pointer  
 $P?P:\{ \}^n$  ha hier, adjektivus zu  
 $P$

## 11 problema:

*uses string*  
push-back (const char&c) { str.push-back(c); }  
outsize () const { return str.size(); } // no signed int

L → seems like coxallate, can we tell chy?  
G or very iron?  
→ fetal iron falls

$\Rightarrow$  kevésbé elterjű felhasználási örökléssel

class safe\_string : public std::string  
 {  
 public:  
 safe\_string(const char \*p) : std::string(p?p:"") {}  
 ~safe\_string() {}  
 };  
C++Guru  
10/2.1

→ tovább hív  
 az std::string  
 (destructor).  
 → is többkell  
 inicializálni.  
 → kérés típuson hagyom  
 megne a

Problema:  
 konstruktorek nem elérhetőek (8-10 konstruktör)  
 de automatikusan meghívhatók a string-hez a műveletek, azokat nem kell.  
 safe\_string s(); } ✓ meghívhatók.  
 s.push\_back('t'); }

## Polimorfizmus (Többalakúság)

Class Base

C++Guru  
10/2

{ public:  
 virtual void f() const  
 {} };

Baseball  
Gázszármaztatás.

## Polimorfizmus (Többalakúság)

Class Base

C++Guru  
10/2

{ public:  
 virtual void f() const  
 { std::cout << "Base::f()" ;  
 }  
 virtual ~Base() {} };

Baseball  
Gázszármaztatás,  
static típus      dinamikus típus

};

Class Der : public Base

{ public:  
 virtual ~Der() {}  
 virtual void f() const  
 { std::cout << "Der::f()" ;  
 } };

Base\* bp = new Base();  
 bp->f();            // Base::f()  
 delete bp;

bp = new Der();  
 bp->f();            // Der::f()  
 delete bp;

bp dinamikus típusa

};

Base b;  
 Base::f() ← b.f();  
 Der d;  
 Der::f() ← d.f();

Destruktörök le fel  
 a Base destruktőre  
 visz + amikor delete  
 Der típus  
 minden konstruktorba nincs hozzáférés!

if (...) {  
 ...  
 }  
 if (...) {  
 ...  
 }  
 else if (...) {  
 ...  
 }  
 else if (...) {  
 ...  
 }  
 else if (...) {  
 ...  
 }  
 ...  
 }

bp → AC;  
 mi fog lehetne?  
 alább-ez a dekonstruktor  
 es esztétikai eredménytelen  
 → fájunk elég új felhívás  
 Cpp 10/3.1  
 10/3.1

class Der : public Base  
 {  
 ...  
 }

→ megírásukban a privat is  
 → A Base kethető része  
 → melegen kethetősgéggel Delagájan  
 kevés öröklődött (A Base ból)  
 a public része

→ def: private öröklések  
 Base public-ut private -kent  
 ezen el,

StartDerBase  
 {  
 ...  
 }

def: public:

Cpp 10/3.2 għad

class Shape  
 {  
 ...  
 }

def: public:

Cpp 10/3.2 għad

class Shape → abżżejt osztójt il-muħekkha objekta u referenci  
 { int color;  
 public:

int get\_color() const  
 { return color;  
 }

virtual ~Shape(); }

(sejgħi kien ġej jaġi minnha )  
 a virtual

→ nincs virtuellis destruktur  
 → Polimorfizmus li ġixx-żejt  
 kien il-lik haqqha

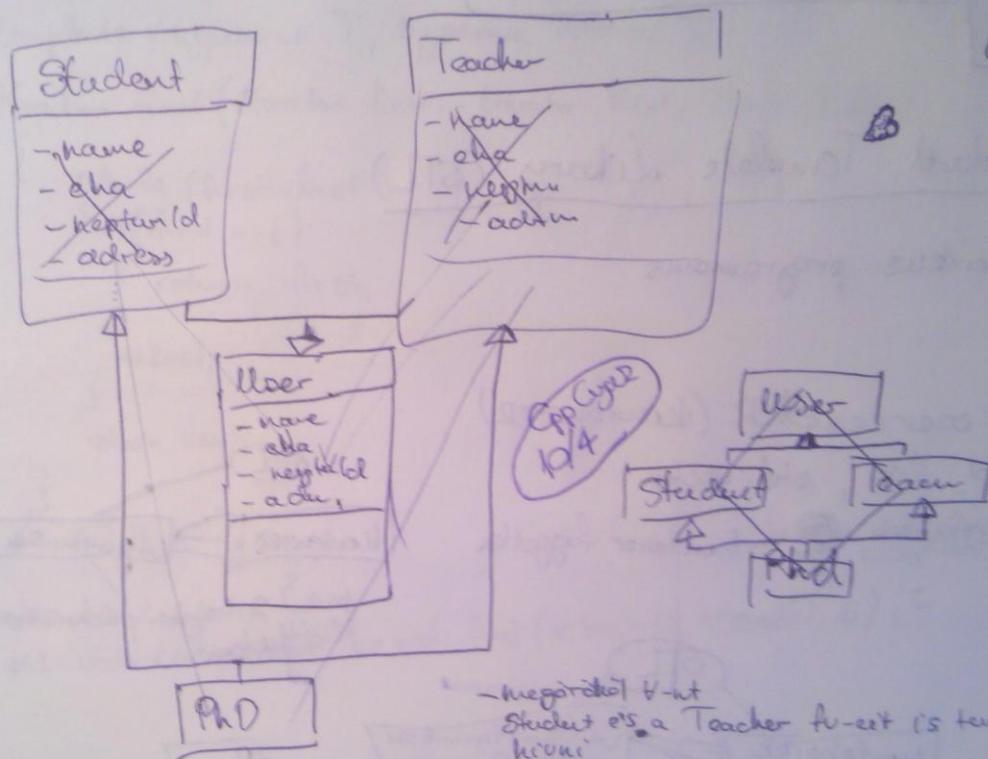
virtual double area() const = 0;  
 ...

//Shape si → /foddl tħalli kien ( nincs! )

Shape \*s = 0;

... s->area();

10/4  
Özgür



- megállító hibát  
Student és a Teacher között is fedje kiuni

- Problémák:  
+ le elég hosszú a szöveg  
van nemzeti (Student.name, Teacher.name)

Megoldás  
→ Virtuális öröklődés

# Szabály

tipusokat töltők parancsosai

① Óban nem vector,

int print f (char\* str);

11.1.1

Cgyar C++

utóra azon belül, ezt a jelölés  
jöhet, ez a jelölés  
⇒ 1 print f

void\* bármilyen adatra mutathat



mátrix

#define MAX(a,b) ((a)<(b)?(b):(a)),

② → szablonok

template <class T>

const T& max (const T&a, const T&b)

mátrix

#define MAX(a,b) ((a)<(b)?(b):(a)),

③ → szablonok

template <class T>

const T& max (const T&a, const T&b)

{ return a<b?b:a;

}

11.1.2  
Cgyar C++

std::cout << max(8, 14);  
cout int

T = cout

↳ lepeladagolja

parameterek dedukció

std::cout << max(7, 2, 6, 5);

T = double

std::cout << max(7, 6);

forrásba hozza

T = cout

T = int

kerül el

önök

Gy

# Speciárium

std::cout << max<double>(6, 7.3);

- <, > operator típus

struct Complex

{

};

11.2.1  
CppCode

Complex a;

Complex b;

Complex c = max(a, b);

T = Complex → hibás operator<  
arra adja a hibát

std::cout << max("abc", "def");

const char[4]

→ példányosít

11.2.2  
CppCode

előző oknál mutató pointerkort addolhat

std::cout << max("abc", "def");

const char[4]

→ példányosít

11.2.2  
CppCode

előző oknál mutató pointerkort addolhat

a < b

"abc" "def" 2 független tömb

compiler figyeli hogyan működik  
használata össze

⇒ Compiler figyel

Nem definiált ezt fox eredményt ha ezek  
független tömbökre mutatnak pointerek.

2 string nem egyszerű hosszú  
⇒ nem fordul le (nem minden típusra)

std::cout << max<std::string>("abc", "def");  
//→ def.

Ganz  
C++

(Startet kein Fehler bearbeitbar)

Class

)

template <typename Iterator, typename Fun>

Fun for-each (Iterator first, Iterator last, Fun f)

{ while (first != last)  
    f(\*first++);  
return f;

Grund Cpp

11/3.1

Iterator f =

- Operator !=
- Operator ++(int) →
- Operator \*
- Copy ctor

\* dereferenzieren  
a(first++) +, -  
etwazt erfordert

Ez pl. Labet  $\approx$  double\*

pointer arithmetik darf nur aufgelöst werden

STL-eo iterator aufzeichnet role

Fun

je weiter  $\approx$  double\*

pointer arithmetik darf nur aufgelöst werden

STL-eo iterator aufzeichnet role

Fun

- Operator ()
- Copy ctor (Fun f) / return f)

konstruktor  $\approx$  konteren element

fu. pointer

Grund Cpp  
11/3.2

functor types

template <(Class T>

class Print

{ std::ostream & os

public:

Print (std::ostream& o): os(o) {}

void operator() (const T& t)

{ os << t << endl;

}

} ;

# Hogyan lehet használni?

11.  
Gyak  
Elt

Print <int> p(std::cerr);

p(5);

p(3);

p("Hello");

→ fordításra kírba → ez nem fog műtően konvertálni

11/4.1  
Copy

Print <double> a(std::cout)

Print <std::vector<int>>

Print <std::list<int>> b(std::cout);

11/4.2  
Copy

Okt.

amoddig nem hozzájönök, a ≈ nincs íre

Print <std::vector<int>>

Print <std::list<int>> b(std::cout);

11/4.2  
Copy

Okt.

amoddig nem hozzájönök a ≈ nincs íre

adding mielőször fordításra kírba

→ a kivájt, nincs nem használható becene

Print <double> a(std::cout);

a(4.4);