Morgan Stanley

Újdonságok az új C++ szabványban - miért várjuk a C++0x-et?

Porkoláb Zoltán <Zoltan.Porkolab@morganstanley.com>

Morgan Stanley

Tartalom

Nagy vonalakban

- Rövid C/C++ történelem
- C++ helye a programozási nyelvek között
- A C++98 és C++03: problémák, hiányosságok
- Főbb újdonságok a C++0x-ben

Kicsit részletesebben

- Concepts
- Move semantics

A C++ történelem

C

- 1972: C (Dennis Ritchie)
- 1978: Kernighan-Ritchie C
- 1989: ANSI C
- 1999: C99

C++

- 1979: C with classes (Bjarne Stroustrup)
- 1983: C++
- 1998: C++ standard ISO/IEC 14882
- 2003: Technikai korrekció
- 2010: C++0x

A C++ helye a programozási nyelvek között

Tiobe (2009.03)

1	1	Java	19.341%	-1.55%	A
2	2	C	15.472%	-0.04%	A
3	3	C++	10.741%	-0.06%	A
4	4	PHP	9.888%	-0.32%	A
5	5	(Visual) Basic	9.097%	-0.69%	A
6	7	Python	6.080%	+1.18%	A
7	8	C#	4.059%	+0.00%	A
8	9	JavaScript	3.678%	+0.75%	A
9	6	Perl	3.462%	-2.09%	A
10	10	Ruby	2.569%	-0.07%	A
11	11	Delphi	2.272%	+0.25%	A
12	14	PL/SQL	1.086%	+0.33%	A
13	12	D	1.076%	-0.37%	A
14	13	SAS	0.792%	-0.13%	A
15	15	Pascal	0.717%	+0.12%	A-
16	21	Logo	0.707%	+0.37%	A
17	27	ABAP	0.658%	+0.42%	B
18	26	RPG (OS/400)	0.646%	+0.40%	B
19	19	Lua	0.491%	+0.12%	B
20	23	MATLAB	0.482%	+0.22%	B

C++ == Hatékonyság + Biztonság

Morgan Stanley

Multiparadigma

Imperatív

- A strukturált programozás teljes készlete

Objektum-orientált

- (Többszörös) öröklődés
- Polimorfizmus

Generatív

- Template-ek, STL
- Metaprogramozás

Funkcionális

- Functorok
- C++0x: Lambda függvények és függvényburkolók

C++0x

- 2005: TR1 könyvtárak
- 2009 március: legutolsó draft
- 2009: életbe lép (?)

Könyvtári bővítések

- Jórészt a boost könyvtárak alapján
- Hiánypótlások (pl. hash, reguláris kifejezések, smart pointer)

Nyelvi bővítések

- Párhuzamos programozás
- Biztonság növelése
- Hatékonyság növelése

Könyvtári bővítések (a TR1 alapján)

- Array
- Tuple
- Hashtáblák (Unsorted associative containers)
- Reguláris kifejezések
- Smart pointerek
- Véletlenszám generátorok
- Függvényburkolók
- Típusinformációk metaprogramozáshoz (type traits)

Nyelvi módosítások

- Általánosított konstans kifejezések
- Inicializálás
- Auto típusok
- Intervallum-alapú for ciklus
- Lambda függvények
- Típusbiztos felsorolási típusok (enum)
- Variadic template-ek
- Párhuzamos programozás elemei
- **Concept-ek**
- **Jobbérték referencia és áthelyező-szemantika**

Concept-ek

A probléma

- A C++ template típustalan
- C++ 98-ban nincsen megszorítás a template paraméterekre (mint pl. Java 5.0 generics)
- Számos hiba csak példányosításkor derül ki
- Váratlan, nehezen értelmezhető hibák

Egy STL példa

```
stl::list<int>  ilist;

Ilist.push_back(1);  ilist.push_back(2);  ilist.push_back(3);

std::sort( ilist.begin(), ilist.end());
```

Concept-ek

```
/usr/include/c++/4.1.3/bits/stl_algo.h: In function ‘void std::sort(_RandomAccessIterator,
_RandomAccessIterator) [with _RandomAccessIterator = std::_List_iterator<int>]’:
templ.cpp:13:   instantiated from here
/usr/include/c++/4.1.3/bits/stl_algo.h:2713: error: no match for ‘operator-’ in ‘__last -
__first’
/usr/include/c++/4.1.3/bits/stl_algo.h: In function ‘void
std::__final_insertion_sort(_RandomAccessIterator, _RandomAccessIterator) [with
_RandomAccessIterator = std::_List_iterator<int>]’:
/usr/include/c++/4.1.3/bits/stl_algo.h:2714:   instantiated from ‘void
std::sort(_RandomAccessIterator, _RandomAccessIterator) [with
_RandomAccessIterator = std::_List_iterator<int>]’
templ.cpp:13:   instantiated from here
/usr/include/c++/4.1.3/bits/stl_algo.h:2360: error: no match for ‘operator+’ in ‘__first + 16’
/usr/include/c++/4.1.3/bits/stl_algo.h: In function ‘void
std::__insertion_sort(_RandomAccessIterator, _RandomAccessIterator) [with
_RandomAccessIterator = std::_List_iterator<int>]’:
/usr/include/c++/4.1.3/bits/stl_algo.h:2363:   instantiated from ‘void
std::__final_insertion_sort(_RandomAccessIterator, _RandomAccessIterator) [with _
```

Concept-ek

A lényeg

templ.cpp:13: instantiated from here
/usr/include/c++/4.1.3/bits/stl_algo.h:2273: error: no match for ‘operator+’ in ‘__first + 1’

A hiba oka

- A list<> konténernek nincsen random-elérésű iterátora
- A hiba példányosítás során jelentkezik, egy standard könyvtári függvény közepén
- C++98-ban nem létezik a sablon-szerződés modell

A megoldás

- Concept a template-paraméterekkel szemben előírt követelmények listája
- Függvény-szignatúrák és segédtípusok

Concept & requires

```
concept LessThanComparable<typename T>
{
      bool operator<(T a, T b);
};

template<typename T> requires LessThanComparable<T>
T min(const T& a, const T& b)
{
   return a < b ? a : b;
}
```

Más formátummal (mint template-ek típusa)

```
template<LessThanComparable T>
T min(const T& a, const T& b)
{
   return a < b ? a : b;
};
```

Többtípusos concept-ek

Kifejezhetjük több típus kapcsolatát

```
concept Convertible<typename T, typename U>
{
    operator U(const T&);
};
```

Használhatunk default értékeket is

```
concept EqualtyComparable<typename T, typename U = T>
{
    bool operator==(T, U)
};
```

Segédtípusok

```
concept InputIterator<typename Iter>  {
  typename value type;
  typename reference;
  typename pointer;
  typename difference type;
  requires Regular<Iter>;
  requires Convertible<reference type, value type>;
  reference operator*(const Iter&);          //…
}
```

Finomítás és túlterhelés

Concept-ek kompozíciója

- Hierarchikus felépítés
- Finomítás: nem öröklés

```
concept InputIterator<typename Iter, typename Value>  : Regular<Value>
{
    Value operator*(const Iter&);       // dereference
    Iter& operator++(Iter&);            // pre-increment
    Iter   operator++(Iter&, int);      // post-increment
};
```

Concept alapú túlterhelés

```
sort ( RandomAccessIterator  begin, RandomAccessIterator  end);

sort ( ForwardIterator  begin, ForwardIterator  end);
```

Concept_map

Szemantikus kapcsolat a concept és a típus között

```
struct Student
{
    std:string name;
    double avg;
};

std::list<Student> l;
std::cout << (*std::min_element(l.begin(), l.end())).name << std::endl;


concept_map   LessThanComparable<Student>
{
    bool operator<(Student a, Student b)
    {
      return a.avg < b.avg;
    }
};
```

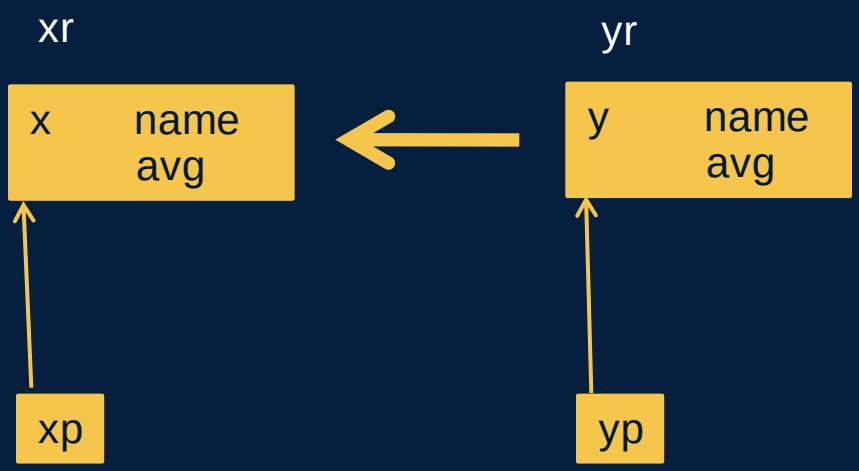
Concept-ek

listconc.cpp: In function ‘int main()’:
listconc.cpp:9: error: no matching function for call to ‘sort(std::_List_iterator<int>, std::_List_iterator<int>)’
/home/lupin/local/conceptgcc/bin/../lib/gcc/i686-pc-linux-gnu/4.3.0/../../../../include/c++/4.3.0/bits/stl_algo.h:2874: note: candidates are: void std::sort(_Iter, _Iter) [with _Iter = std::_List_iterator<int>] <where clause>
listconc.cpp:9: note: no concept map for requirement ‘std::MutableRandomAccessIterator<std::_List_iterator<int> >’

Jobbérték referencia és áthelyező szemantika

A C++ másoló szemantikára épül

```
class Matrix
{
   std::string  name;
   double  avg;
};

Student   x,  y;

x = y;


Student   *xp = &x,  *yp = &y;


Student  &xr = x,  &yr = *yp;
```



Felhasználói másoló műveletek

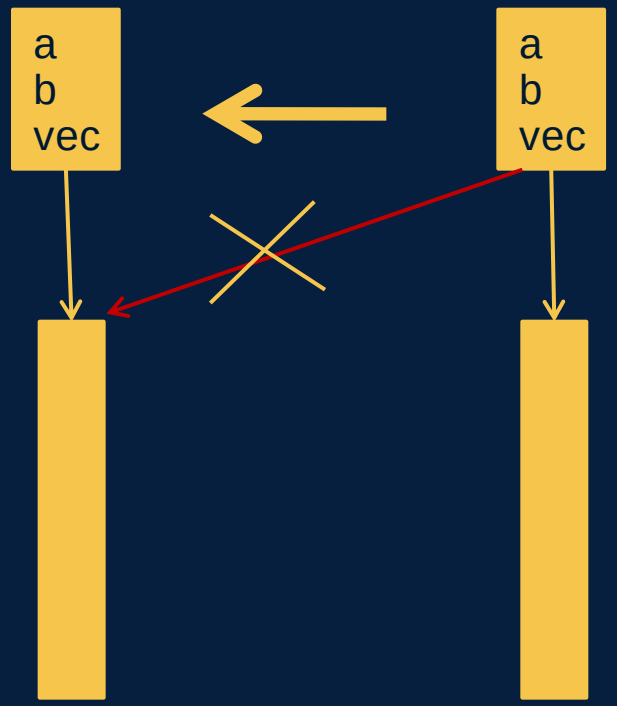
Egy nem POD osztály

Nem másolható byte-onként

```
class  Matrix::operator=(const Matrix &rhs)
{
    int a, b;
    double  *vec;
};
```

Felhasználó értékadó operátora

```
Matrix&  Matrix::operator=(const Matrix &rhs)
{
    if ( this != &rhs )
    {
        delete [] ptr;
        a = rhs.a; b = rhs.b;
        vec = new double[a*b];
        copy (rhs);
    }
    return *this;
};
```



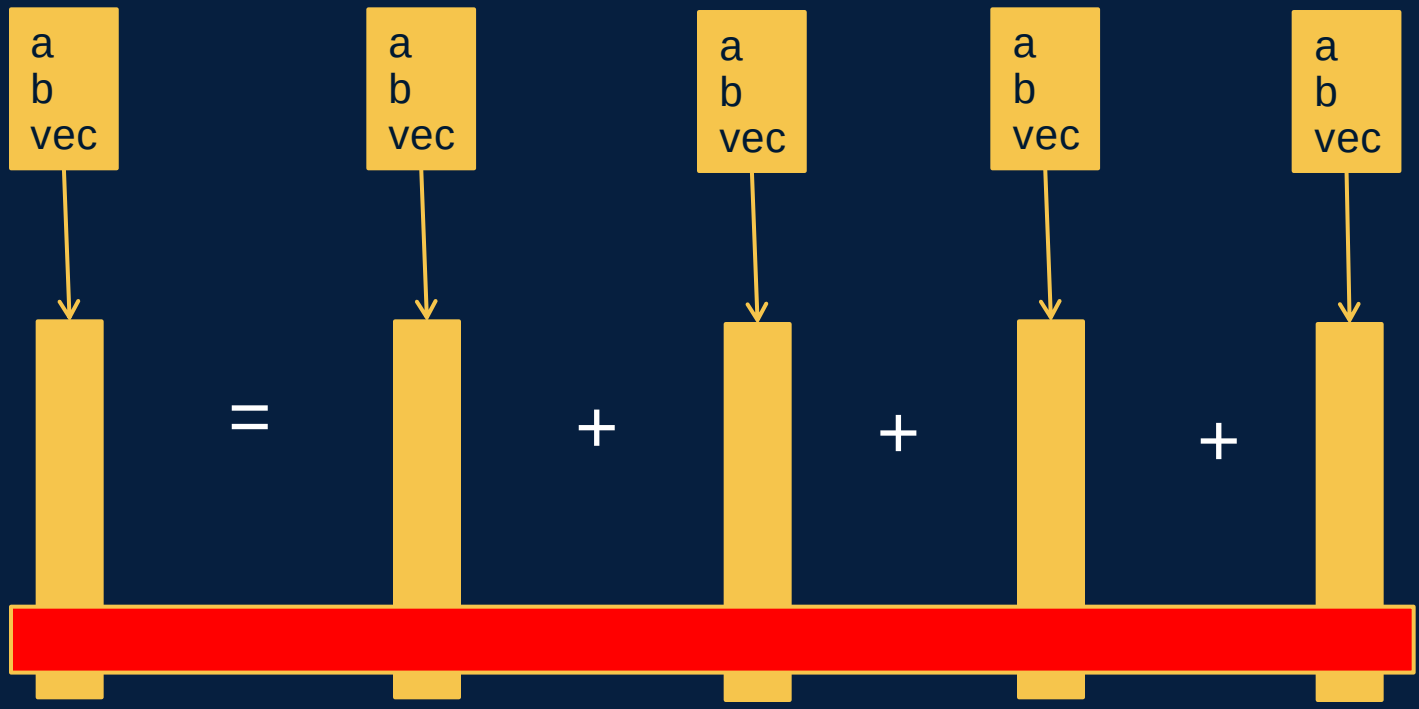
Egyszerű mátrix műveletek

Elv: interfész és implementáció elválasztása

Túlterhelt operator+

Felhasználói értékadó operátor

A = B + C + D + E;



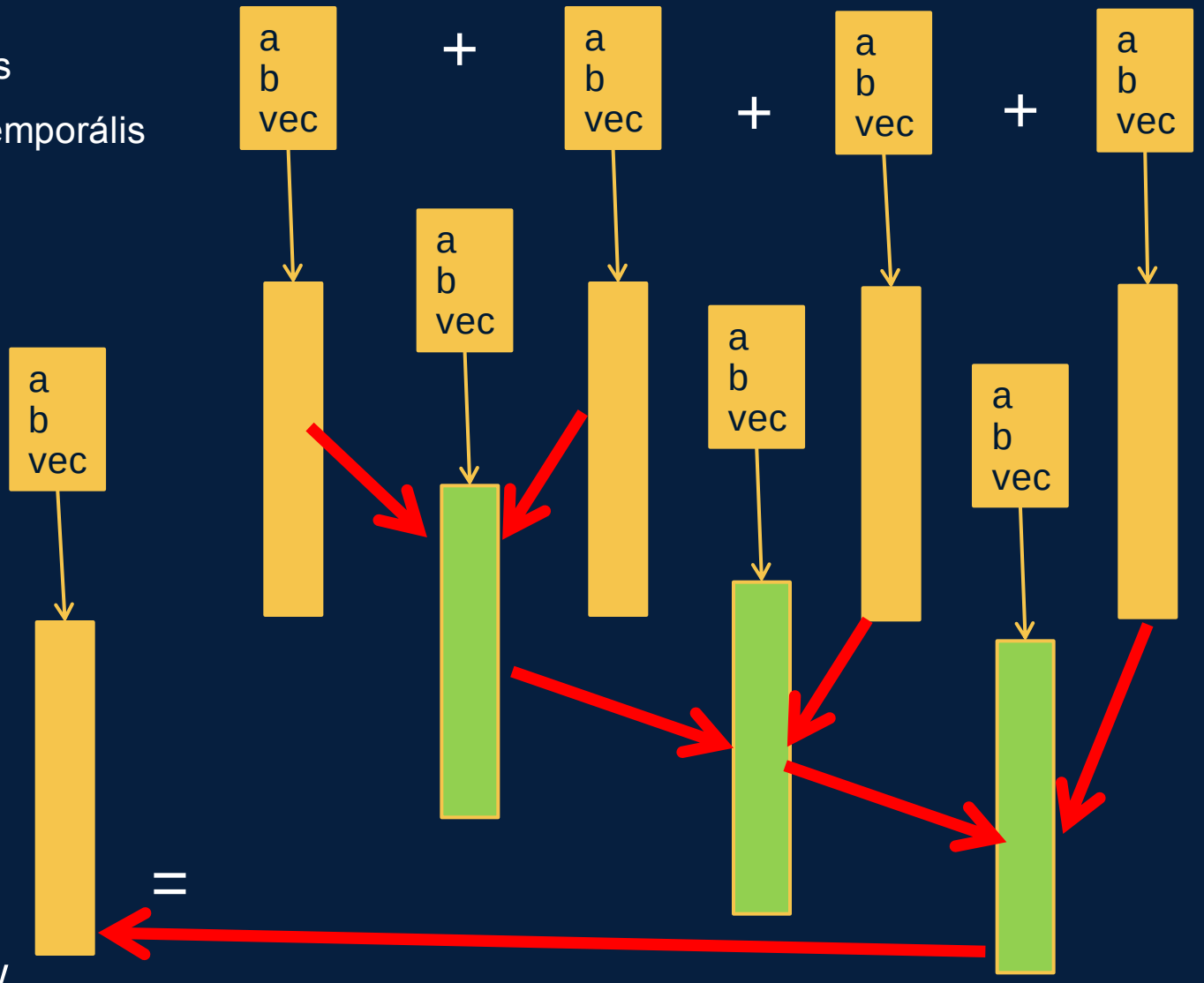
Egyszerű mátrix műveletek

Gyakorlat

Felesleges és

költséges temporális

objektumok



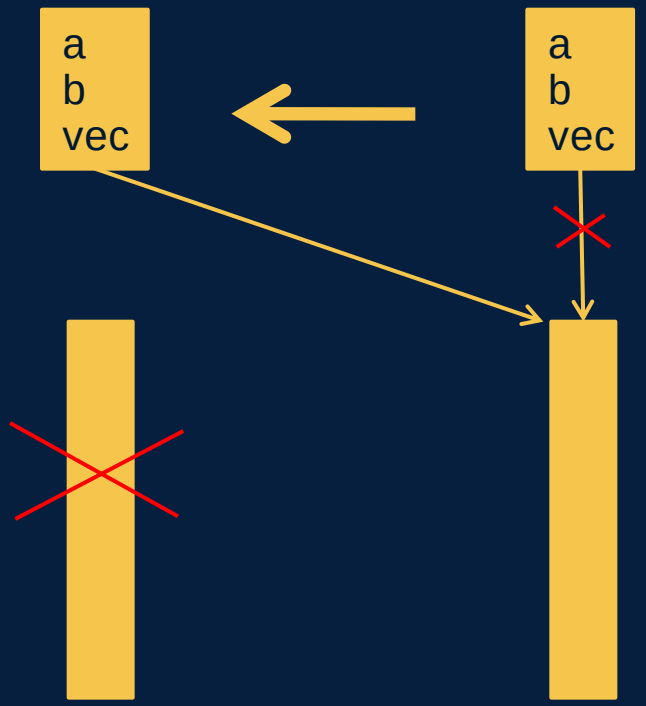
Jobbérték referencia

Temporálisokhoz köt

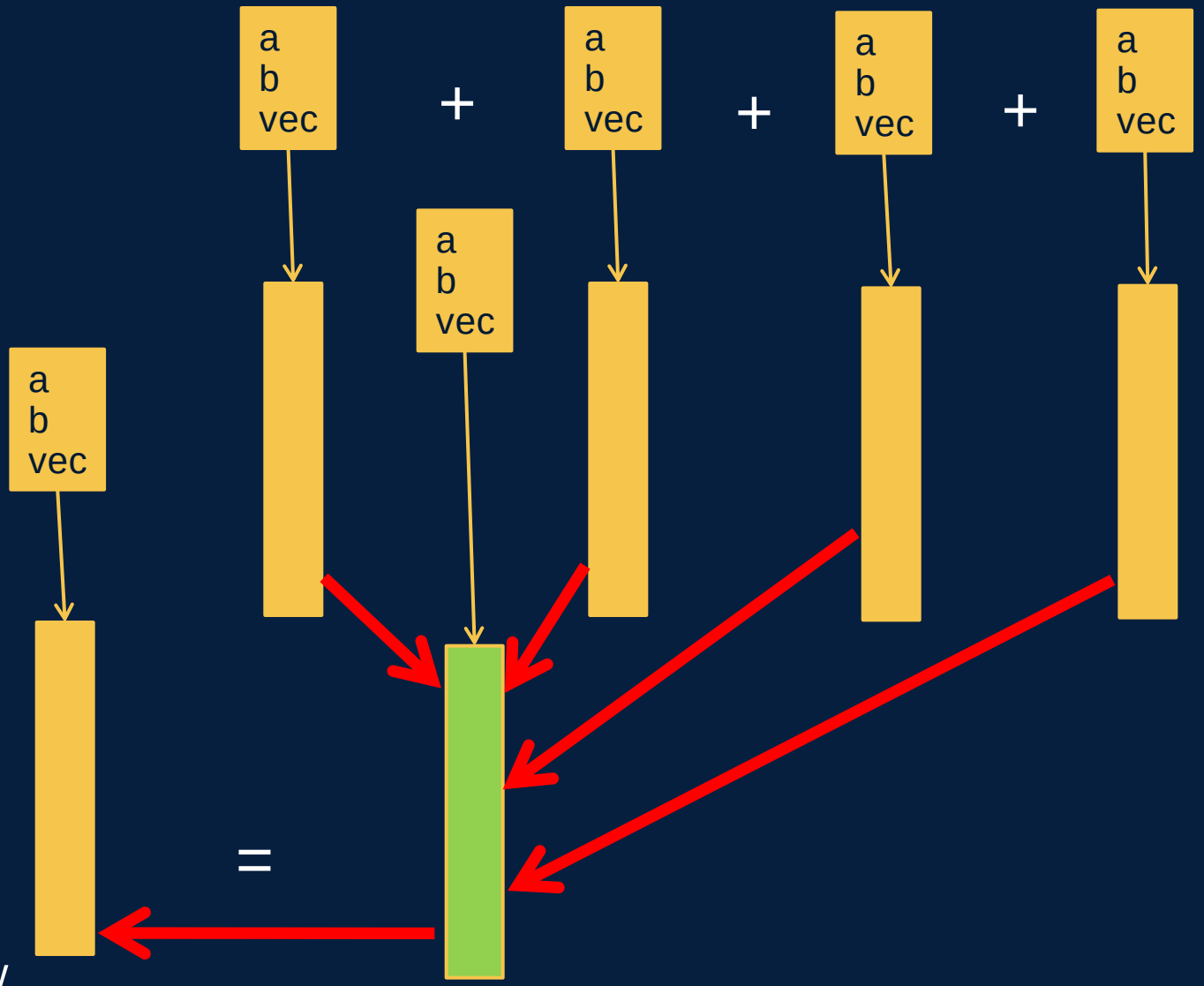
```
int i = 5;

int  &leftref  =  i;
const int  &cref = 3+4;
Int  &&rightref   = 3+4;
```

Felhasználó áthelyező operátora

```
Matrix&  Matrix::operator=(const Matrix &&rhs)
{
    if ( this != &rhs )
    {
        delete [] ptr;
        a = rhs.a; b = rhs.b;
        vec = rhs.vec;
        rhs.vec = 0;
    }
    return *this;
};
```



Áthelyezés szemantika



Morgan Stanley

Morgan Stanley's Global Presence

600 Offices in 32 Countries. **Over 58,000 Employees.** **6,000 in Technology Worldwide**

Morgan Stanley

Morgan Stanley

Köszönöm a figyelmet!

Morgan Stanley