A Programozás Alapjai 2
Objektumorientált szoftverfejlesztés

Dr. Forstner Bertalan
forstner.bertalan@aut.bme.hu


I ♥
ZH
ASTRA
H
SVAJC-1

C++ IO, operátorok túlterhelése

Streamek

- C: stdin, stdout, stderr
- C++: cin, cout, cerr

- Lehetnek input és output streamek
- istream csak olvasható, ostream csak írható.

Streamek

- C: stdin, stdout, stderr
- C++: cin, cout, cerr

- Lehetnek input és output streamek
- istream csak olvasható, ostream csak írható.
- << és >> operátorokkal lehet őket írni/olvasni, illetve egymás után fűzhetőek ezek a műveletek.
- Példa

Bevezető példa

```
cout << "Enter an int and a double:" << endl;
int i;
double d;
cin >> i >> d;
cout << "What you entered is: " << i << ', ' << d << endl;
```

4 bit:

- good – nincs hiba
- eof – file vége
- bad – adatvesztés történt
- fail – formátumhiba

4 bit:

- Beállítása (pl. cout streamen):
  > cout::clear(ios::failbit);
- Lekérdezése (pl. cin streamen):
  > cin.good()
  > cin.eof()
  > cin.bad()
  > cin.fail()
    - ez akkor is, ha a bad be van állítva.
    - Ilyenkor további írás-olvasás nem történik a clear-ig.

Visszatérési érték

- istream és ostream értéke a !fail() értéket adja vissza, így kevesebbet kell gépelnünk

- Példa: fájlmásolás

Példa: fájlmásolás

```
ifstream from("C:\\temp\\adat.txt");
if (from)
{
   ofstream to("C:\\temp\\adat2.txt");
   if (to) {
      char ch;
      while (from.get(ch)) {
         to.put(ch);
         cout << ch;
      }
   }
   if (!from.eof() || !to)
      cerr << "Varatlan hiba" << endl;
}
```

Formázás

```
printf("(%8.2f,%8.2f)\n", x, y);
```

- Itt a mezőszélesség 8, és két tizedesjegyet írat ki. Mi a megfelelője ennek C++-ban?
- Hogy néz ki C++-ban? Példa

Formázás

```
printf("(%8.2f,%8.2f)\n", x, y);
```

- Itt a mezőszélesség 8, és két tizedesjegyet írat ki. Mi a megfelelője ennek C++-ban?
- Hogy néz ki C++-ban? Példa

```
cout << '(' << setprecision(2) << setiosflags(ios::fixed) << setw(8)
   << x << ',' << setw(8) << y << ')' << endl;
```

Manipulátorok

- A példában *setprecision*, *setiosflag*, *setw* és *endl* úgynevezett IO manipulátorok
- Van, amelyiknek paramétere van (*setprecision*), van, amelyiknek nincs (*endl*)
- Ezek az *iomanip* állományban találhatók.

Miket formázhatunk?

- Mezőszélesség
- A kitöltő karakter (alapértelmezett: space)
- Igazítás
- Az egész számok számrendszere (decimális, hexadecimális, oktális)
- A lebegőpontos számok formátuma (fix, tudományos, általános)
- Mutassa-e: + jel, helykitöltő nullák, az egész számok számrendszerének alapja
- Kis- vagy nagybetű a normálalak E-jében, vagy a hexadecimális számjegyekben

Hogyan formázhatunk?

- Kétféle módon érhetjük el a hatást:
  > Az output stream tagfüggvényeivel
  > Manipulátorokkal

- Példa (számformátum):

```
cout <<setiosflags(ios::fixed);
```

- vagy:

```
cout.setf(ios::fixed);
```

manipulátor	ios tagfüggvény
`setfill`	`fill`
`setw`	`width`
`setprecision`	`precision`
`setiosflags`	**`setf`**
`resetiosflags`	**`unsetf`**
`hex`	
`oct`	
`dec`	

Teljes sor olvasása, whitespace kezelés

- Getline függvény olvas be teljes sort egy stringbe (nemcsak whitespace-ig)

```
std::cout << "Mi a teljes neved: ";
std::string name{};
std::getline(std::cin >> std::ws, name);
std::cout << name;
```

- A ws manipulátor a whitespce karaktereket ignorálja
- *std::cin >> std::ws* – a kifejezés értéke a *cin* lesz, de már a bevezető whitespace-ek nélkül.

Jelzőbitek

- A jelzőbiteket a *setiosflags*/*resetiosflags* manipulátorokkal állíthatjuk
  > illetve a setf/unsetf tagfüggvényekkel
- Mindaddig megtartják beállításaikat, ameddig meg nem változtatjuk őket
  > Kivéve a mezőszélesség

ios::___	**jelzőbit jelentése**
left	balra rendez
right	jobra rendez
internal	sign left, remainder right
dec	decimal base
hex	hex base
oct	octal base
(no)showbase	show integer base
showpos	show + sign
(no)uppercase	uppercase E, X, and hex digits A ... F
fixed	fixed floating point format
scientific	scientific floating point format
(no)showpoint	show trailing decimal point and zeros

Operátorok túlterhelése

Operátorok

- a+b, a=b, a==b, new, delete
  > +, =, ==, new, delete, … mind operátorok

Operátorok

- a+b, a=b, a==b, new, delete
  > +, =, ==, new, delete, … mind operátorok
  > Úgy is nézhetjük, mint egy függvényhívás, csak más a szintaktika
- c=a+b;

Operátorok

- a+b, a=b, a==b, new, delete
  > +, =, ==, new, delete, … mind operátorok
  > Úgy is nézhetjük, mint egy függvényhívás, csak más a szintaktika
- c=a+b;
  > c=operator+(a,b);

Operátorok

- a+b, a=b, a==b, new, delete
  > +, =, ==, new, delete, … mind operátorok
  > Úgy is nézhetjük, mint egy függvényhívás, csak más a szintaktika
- c=a+b;
  > c=operator+(a,b);
  > operator=(c, operator+(a,b))
  //Persze ezek pl. int-re nem működnek így!

Operátorok túlterhelése

- A függvények túlterhelhetők: többet is írhatunk ugyanazzal a névvel, csak legyen más az argumentum lista (a visszatérési érték nem megkülönböztető).

Operátorok túlterhelése

- A függvények túlterhelhetők: többet is írhatunk ugyanazzal a névvel, csak legyen más az argumentum lista (a visszatérési érték nem megkülönböztető).
- Az operátorokat is túl tudom terhelni a következő feltételekkel:
  > jelölés (név) és argumentum lista alapján legyen egyértelmű
  > legalább az egyik argumentum nem beépített típus kell legyen (pl. két int összeadását nem tudom megváltoztatni)
  > Új operatátort nem lehet bevezetni: pl. **-ot.
  > Ha felül is definiálom: marad a precedencia szint és az asszociativitás.
  > Nem felüldefiniálható a . struktúramező elérés, a :: scope, a ?:

Szintaktika

- Hasonló a függvényekhez, csak a neve speciális.
- visszateresi_típus operator<opjel>(arg lista) {…}

- Példa: komplex számokat ábrázoló osztály

Operátor tagfüggvények

- Az objektum tud magára vigyázni, és az adatot és a rajta végezhető műveletet egységbe szeretnénk zárni

==> Jó lenne, ha az operátor tagfüggvény lenne.

- Lehet is, ha csak lehet, így írjuk meg.
- Az ilyen tagfüggvények esetén az operátor 1. operandusa mindig az az objektum, amire meghívtuk.
- Példa

Egyargumentumú operátorok

Kifejezés	Operátor (@)	Taggfügvény	Globális függvény
@a	+ - * & ! ~ ++ --	A::operator@()	operator@(A)
a@	++ --	A::operator@(int)	operator@(A, int)

Kétargumentumú operátorok globális függvénnyel

Kifejezés	Operátor (@)	Taggfügvény	Globális függvény
a@b	+ - * / % ^ & | < > == != <= >= <<	A::operator@(B)	operator@(A, B)

További kétargumentumú operátorok

Kifejezés	Operátor (@)	Taggfügvény	Globális függvény
a@b	= += -= *= /= %= ^= &= |= <<= >>= [ ]	A::operator@(B)	-
a(b, c...)	()	A::operator()(B, C...)	-
a->b	->	A::operator->()	-

operator= példa

```
Person& operator=(const Person& theOther) {
	if (this != &theOther) { //Fontos, különben hiba lenne
		delete[]  name;
		name = nullptr;
		name = new char[strlen(theOther.name) + 1];
		strcpy_s(name, strlen(theOther.name) + 1, theOther.name);
	}
	return *this;
}
```

Összefoglalás

- C++ I/O
  - > cin, cout, cerr
  - > Barátunk, a << és >>
  - > Manipulátorok
- Operátorok túlterhelése
  - > Globális függévnyként
  - > Tagfüggvényként
  - > Figyeljünk a visszatérési értékre
- nagyZH felkészítés – video példafeladattal!