Översikt

- const
- operatorer inkl tilldelning
- konverteringar

Const - ett underskattat nyckelord

- const int glif = 5;
- const char *pc = "hej"; // pekar på konstant
- char *const cp = "hej"; // konstant pekare
- const char * const cpc="hopp";
- const char* peek(int i) { return hidden[i]; }

Const i klasser

```
class K
{
public:
  int uttag(int bel);
  void get_saldo() const;// Får ej ändr medl var
}
const K guldReserv;
guldReserv.uttag(5000); // Kompil. fel
int i= guldReserv.get_saldo(); // OK
```


Operatorer & precedens(prioritet)

1. Person::k, ::savefile
2. k.namn, p->saldo, i[5], fkn(), (3), C++, i--
3. sizeof(v), ++i, --h, ~0, !0, -3, +37, &mig, *p, new, delete, delete[], (int)3.14
4. k.*mfp(), p->*mfp()

Operatorer & precedens-2

5. 3*4, 7/0, 1234%256

6. 3+4, 1-8

7. cout << i, j >> 8

8. a<0, <=, >, >=

Operatorer & precedens-3

9. a == b, a != 3

10. a & 0x7f
11. si ^ sa
12. a | 4

13. i<5 && a[i]
14 a || b

Operatorer & precedens-4

15. a ? 1 : 0

16. =, *=, /=, %= +=, -=, <<=, >>=, &=, |=, ^=

17. throw "Bad no"

18. 3,4

Överlagring av operatorer

- För uttryck med egna typer
  - Kan ej omdefinieras för inbyggda typer
- De flesta (dock ej :: . .* ? :)
- Inga nya operatorer t ex < >, **
- Samma precedens & associativitet
- Samma antal operander (unär/binär)
- 2 sätt att överlagra

Operator som medlem

```
class B
{ public:
  B operator+(B& b);
}
B B::operator+(B& b)
{
  B res;
  res.saldo = saldo + b.saldo;
  return res;
}
deras = hans + hennes;
deras = hans.operator+(hennes);
```


Operator som friend

```
class B
{ public:
friend operator+(B&,B&);
}
B operator+(B& b1, B& b2)
{
  B res;
  res.saldo = b1.saldo + b2.saldo;
  return res;
}
deras = hans + hennes;
deras = operator+(hans,hennes);
```

Som Medlem Som Friend

- Föredras
- = [] () ->

- Undviks
- Utskriftsop << >>
- Typkonv; om vänster operand ej är objekt: 1+mitt 1.operator+(mitt)

Tilldelningsoperatorn

- Default: grund kopiering (medlemsvis)
- Vi vill ofta ha djup (om vi använder pekare)
- Undvik tilldelning av sig själv

Tilldelning: Ett litet vektorexempel

```
class V
{
public:
  V(int sz);
  operator[](int i);
private:
  int storlek;
  int* vk;
}
V A(5), B(5);
A[0] = 15;
B = A; // ??
```


Ett litet vektorexempel (forts)

```
V& operator=(V& v2)
{
  if (this != &v2) // undvik A=A
  {
    delete [] vk;
    vk = new int[storlek = v2.storlek];
    for (int i=0; i<storlek;i++)
    {
      vk[i] = v2[i];
    }
  }
  return *this;
}
```

Initieringskonstruktorn

- Används i 4 fall;
  - V b = a; // Initiering av objekt från ett annat
  - void fkn(V b) {}; fkn(a); Parameteröverföring
  - V fkn() { V t; return t; } // Init av returvärde
  - d=a+b*c; // kompil. skapar temporär t=b*c

Övrigt om operatorer

- operator++() är prefix
- operator++(int) är postfix(dummyvariabel)
- Betyder == samma obj eller värde?

Konverteringar

- implicit / explicit(cast)
- Klass till fördef:
  - operator double() {} i klassen
- Fördef till klass
  - konstruktor med parameter: K(double v)
- Klass till klass
  - konstruktor med obj som parameter: K(P& p)