Introduktion till C++

- Grundläggande C++
- Ett programexempel
- Fördefinierade datatyper
- Nya reserverade ord
- Viktiga skillnader mot C
- Nya streams I/O ersätter stdio
- Dynamiskt minne
- Referensvariabler
- Klasser

Översiktsbild (forts.)

- Ett exempel
- Implementering
- Användning
- Initiering
- Tilldelning
- Med pekare
- Konstruktor
- Klass med konstruktor

Grundläggande C++

- Streamsbiblioteket
- const istf #define
- Parametrar måste definieras (som ANSI)
- Nya kommentarer med // till radslut

Ett programexempel

Ränteberäkning

Streamsbiblioteket

```
#include <iostream.h>
const double ranta = 8.5;
```

Används istället för #define

```
int rante_ber(int summa)
{
  int resultat;
  resultat = summa * (ranta/100);
  return resultat; // Den beräknade räntan
}

void main()
{
    int belopp;
    cout << "Ange belopp";
    cin >> belopp;
    cout << "Räntan på beloppet blir " <<
    rante_ber(belopp) << endl;
}
```

"Strömmar" ut till vänster

Inget &

nyrad

Fördefinierade datatyper

- char, short int, int, long int
- float, double
- signed/unsigned

Nya reserverade ord

```
class
public      private     protected
friend      virtual
operator    inline
this        new         delete
template
try         catch       throw
```

Viktiga skillnader mot C

- Funktioner måste deklareras innan anrop
- Tidigare global `const` är lokal i filen
- Typen på en teckenkonstant 'a' är char, ej int
- Kommentarer skrivs också med //
- Variabler kan definieras inne i koden

Strömmar

- Typsäkert
- Snabbt
- Utökbart
- Objekt-orienterat
- Även för filer och med strängar

Standardströmmar

- cin - tangentbordet
- cout - skärmen
- cerr, clog - för felmeddelanden

Nya streams I/O ersätter stdio

```
// Utmatning till skärm:
cout << ”Hej!”;

int i; cin >> i; // Läs från tangentbord

// manipulatorer:
cout << setw(10) << i;
```


Operationer på strömmar

- #include <iostream.h>
- stream << arg; stream >> arg;
- stream.read(char *buffer, int length);
- stream.write(char *buffer, int length);
- stream.getline(char *line, int len, char delim = '\n');
- stream.get(char& ch);
- stream.put(char ch);
- char stream.peek();

Manipulatorer

- Skjuts in i strömmen av tecken
- Både vid inläsning och utskrift
- #include <iomanip.h>
- endl - ger ny rad
- setw(int w) - Fältbredd
- setprecision(int p) - Antal decimaler
- leftjust, rightjust -vänster/högerställt
- setiosflags(ios::scientific) - med exponent

Dynamiskt minne

- new och delete ersätter malloc och free:

```
person *pp = new person;
delete pp;
```

Eller (en vektor):

```
int *ip = new int[100];
delete [] ip;
```


Referenser

- Låter kompilatorn hantera adresser
- Ungefär som Pascals VAR parametrar

```
swap(int& i, int& j)// call-by-reference
{
  int t = i;
  i = j;
  j = t;
}
int a=2,b=4;
swap(a,b); // obs, inga &
cout << a << ' ' << b; // skriver ut 4 2
```


Referensvariabler

```
int i;
int &ir=i; // alias för samma
  variabel
i=3;
ir=5;
cout << i; // skriver ut 5
```


Att returnera en referens

- Anta att vi har en funktion vekt:

```
int& vekt(int i)
{
  static int v[100];
  return v[i];
}
```

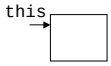
- Då kan vi göra:

```
int k; k = vekt(5);
vekt(3) = 4; // tilldelning
```


Att returnera sig själv, *this

- Om man har en medlemsfunktion:

```
Bankkonto& Bankkonto::ins(int belopp)
{
  if (belopp > 0) saldo += belopp;
  return *this;
}
```



- Så kan man kedja anropen:

```
Bankkonto bk;
bk.ins(500).ins(300).ins(100);
```

Returnerar ett bankkonto

Funktioner

- Prototyper krävs
- Överlagring (samma namn, olika parametertyper)
- Inline expansion (ej loopar eller rekursiva funktioner)
- Namnmangling ger typsäker länkning
- Defaultvärden (från höger)
- Kan ge operatornamn

Några exempel på funktionsdeklarationer

- int max(int i,int j);
- double max(double f, double g); överlagrar föregående
- inline int max(int i,int j); expanderas och ger snabbare kod, typsäkrare än makron
- void initport(int baud, int bits=8);
- Bankkonto operator+(Bankkonto a, Bankkonto b);

Vänner, friends

- Används när vanliga funktioner behöver nå klassmedlemmar
- Klassen talar om vem man litar på

```
Class Bankkonto
{
public:
friend ostream& operator<<(ostream& os,
                           Bankkonto& bk);
private:
int saldo;
};
```


Utskriftsoperatorn

- Bör vara vän för att man inte ska behöva ändra i standardbiblioteket

```
ostream& operator<<(ostream& os, Bankkonto& bk)
{
  os << bk.saldo; // Når känslig info
  bk.saldo += 1000; // späd på veckopengen
  return os;
}
Bankkonto mitt;
mitt.ins(300);
cout << mitt; // Har trevlig bieffekt :-)
```


Filhantering

- Konstruktor/destruktor öppnar/stänger filen
- ifstream, ofstream, fstream

```
#include <fstream.h>
int get_antal()
{
  ifstream datafil(”PROV.TXT”);
  int i;
  datafil >> i;
  return i;// destruktor anropas
}
```


Felhantering

- if (cin >> x)
- while (!cin.eof())
- if (!cin) cerr << ”Fel vid inmatning”;