Relationer mellan klasser
Operatoröverlagring

Och lite diverse

Innehåll

- Repetition och frågor om tidigare delar
- Mer om klasser: const, referens, static, friend, explicit
- Mer om klasser: arv (är), composition (vital del av), aggregation (umbärlig del av), association (känner till)
- Mer om klasser: operatorer
- Funktionsobjekt !!!
- C-bibliotek: cstdlib, cmath, cctype, ctime
- Mer om STL: random, regex, string
- C-strängar, C-arrayer (inbyggda arrayer), (Pekare)
- Kommandoradsargument
- Enkla templates

Const- och referensmedlemmar

- Referenser och konstanter är speciella. Varför? De måste initieras direkt vid deklaration, och detta går inte ändra senare.
- Hur fungerar detta om referensen eller constanten är medlem av en klass?
- Initieringslista måste användas!

Rita! Förklara!

Static-medlemmar

- Normalt får varje instans av klassen en egen kopia av alla medlemsvariabler.
- Normalt skickas en pekare (this) till instansen med när en medlemsfunktion anropas.
- En medlemsvariabel eller medlemsfunktion kan vara ”static”.
- Medlemsvariabeln är då gemensam för alla instanser. Finns aldrig mer än en.
- Medlemsfunktionen får inte en referens till instansen vid anrop. Kan då inte använda några icka-statiska medlemmar. Kan å andra sidan användas utan att ha någon instansvariabel.

Const medlemsfunktioner

- Lovar dyrt och heligt att de inte ändrar på någon medlemsvariabel.
- Kan därmed anropas på instansvariabler som är konstanta.
- Kompilatorn kontrollerar detta NOGA!
  - Vanligt och jobbigt fel att ”const” är det enda som skiljer för att det skall fungera.
  - Ta för vana att alltid skriva const när det går.
- Lös problem med const genom att:
  - Göra medlemsfunktioner till const
  - Använda const_iterator
  - Göra parametrar till const&
- Ta inte bort ”const” för att få det att fungera även om det är frestande. Det är bara en tillfällig lösning medan du väntar på hjälp!

friend

```
class Boy
{
public:
  friend void doFunnyStuffTogether(Boy b, Girl g); // Need to be friend of Boy?
private:
  Boystuff private_part;
};

class Girl
{
public:
  friend void doFunnyStuffTogether(Boy b, Girl g); // Need to be friend of Girl?
private:
  Girlstuff private_part;
};

void doFunnyStuffTogether(Boy b, Girl g)
{
  // Är inte en medlemsfunktion av någon klass, så för att
  // få tillgång till privata delar av en klass måste den
  // klassen ha med en friend-deklaration som ovan.
}
```

explicit

```
class Gay
{
  public:
    Gay(Boy); // make explicit ??
    void doFunnyStuffTogether(Gay a);
};

int main()
{
  Boy hetero, bi;
  Gay gay(bi);
  gay.doFunnyStuffTogether(hetero); // ??
}
```

Relationer mellan klasser

- Arv (Inheritance)
  - Relationen ÄR. En katt ÄR ett djur.
- Composition
  - Relationen VITAL DEL AV. Skelett är en vital DEL AV en ko. Tas skelettet bort är det inte en ko längre. Men kanske en blodig biff?
- Aggregation
  - Relationen UMBÄRLIG DEL AV. Fingrar är en umärlig DEL AV en människa. Tas (några) fingrar bort är det fortfarande en människa. Kanske funktionshindrad, men människa.
- Association
  - Relationen KÄNNER TILL. En människa känner till kreditkort och kan (kanske) använda det.

Någon som minns?

- Programming in the language versus programming into the language
  - Anpassa inte dina tankar till språket.
  - Utryck dina tankar med hjälp språket!
  - Koden skall vara en spegling av hur du tänkt.
- Hur hjälper C++ till?

Arv

```
class Animal
{
public:
  Animal(bool petable)
   : m_petable(petable) {}

  bool dangerous();

protected:
  bool m_hungry;
private:
  bool m_petable;
};
```

```
class Cat : public Animal
{
public:
  Cat() : Animal(true) {}
};

class Lion : public Animal
{
public:
  Lion() : Animal(false) {}
};

class Circus_Lion ... ?
```

Terminologi

class Cat : public Animal {...};

Animal är basklass.
Cat är en klass härledd från basklassen Animal.
Cat är en subklass till Animal.
Animal är superklass till Cat.

Public, protected, private

- Publikt arv (class Cat : public Animal)
  - medlemmar behåller samma åtkomst.
- Skyddat arv (class Cat : protected Animal)
  - publika medlemmar från basklassen blir skyddade i härledda klassen
- Privat arv (class Cat : private Animal)
  - medlemmar från basklassen blir privata i subklassen
  - basklassen blir oåtkomlig, composition bättre?
- Privata medlemmer från basklassen är alltid oåtkomliga i den härledda klassen

Composition and Aggregation

```
class Cow
{
public:
  Cow();

private:
  Skeleton m_skel; // Composition
  Nipple m_nip[4]; // Aggregation
};
```

Associationer

```
int main()
{
  Boy bertram;
  Wife hilma(bertram); // hilma (Wife) känner till bertram (Boy)

  cout << bertram.mode() << endl; // deprimerad

  Girl asta;
  bertram.doFunnyStuffTogether(asta); // bertram (Boy) känner till asta (Girl)

  cout << bertram.mode() << endl; // glad
  cout << hilma.mode() << endl; // arg och ledsen

  Credid_Card visa(bertram); // Ett kreditkort ägs av någon...
  Store hm;
  asta.workAtStore(hm); // Girl känner till Store
  hilma.goShopping(hm, visa); // Wife känner till Store och Credit_Card

  cout << hilma.mode() << endl; // skadeglad
  cout << asta.mode() << endl; // arg och ledsen
  cout << bertram.mode() << endl; // deprimerad
}
```

Operatorsökning

- Vad händer om vi försöker addera två variabler av klasstyp?

```
Girl hilda;
Boy bertram;
?? = hilda + bertram; // Fall 1 ??
?? = bertram + hilda; // Fall 2 ??
```

- Det blir kompileringsfel i fall 1 efter sökning av:
  - Girl::operator+(Boy)
  - operator+(Girl, Boy)
- Det blir kompileringsfel i fall 2 efter sökning av:
  - Boy::operator+(Girl)
  - operator+(boy, Girl)

Operatoröverlagring

- Vad händer om vi skriver någon av sökta funktioner?

```
Couple operator+(Girl g, Boy b)
{
  return Couple(g, b);
}
```

- hilda + bertram går nu att skriva! Lämplig operator+() anropas!
- Resultatet bestäms av returvärdet från operator+

Medlem eller fri?

- Antag att a och b är av klasstyp A och B.
- Om operatorfunktionen är medlem i A gäller för a + b:
  - anropet a.operator+(b)
  - a nås via nyckelodet ”this”
  - b nås via första parmeter
- Om operatorfunktionen är en fri funktion (inte medlem i någon klass) gäller för a + b:
  - anropet operator+(a,b)
  - a nås via första parameter
  - b nås via andra parameter
- Om operatorfunktionen är medlem i B kan a + b inte utföras då vänstersidan av operatorn då måste vara av typ B för att operatorfunktionen skall hittas.

Operatorfunktioner

- Några tänkbara operatorfunktioner för klass A. B, C, D representerar någon annan datatyp.

```
A A::operator+(A const&)
A A::operator+(B const&)
A operator+(B const&, A const&)

ostream& operator(ostream&, A const&)
istream& operator(istream&, A&)

bool operator<(A const&)
bool operator==(A const&)

C& operator[](int)
D operator()(???)
```

Funktionsobjekt

- Diskutera exempel: Summera 10 heltal.
- Lös på olika sätt!

Använda C-funktioner

- C-biblioteket har många delar som fortfarande är användbart i C++.
- En given inkluderingsfil <header.h> inkluderas i C++ genom att skriva <cheader> istället:
  - <cstdlib> istf <stdlib.h>
  - <cctype> istf <ctype.h>
  - <cmath> istf <math.h>
  - <ctime> istf <time.h>

<cstdlib>

```
struct div_t {
  int quot;
  int rem;
};
struct div(int numer, int denom);
int abs(int value);

int rand();
void srand(unsigned int seed);

void exit(int status);
```

<cctype>

```
int tolower(int)
int toupper(int)

int isalpha(int)
int isalnum(int)
int isblank(int)
int iscntrl(int)
int isdigit(int)
int isgraph(int)
int islower(int)
int isprint(int)
int ispunct(int)
int isspace(int)
int isupper(int)
int isxdigit(int)
```

<cmath>

```
#define M_PI  3.1415...
double sin(double)
double cos(double)
double pow(double)
double sqrt(double)
double exp(double)
double floor(double)
double ceil(double)
double floor(double)
double round(double)
```

<ctime>

```
// Sek. sedan kl 00:00 1970-01-01
time_t time(nullptr)

// Ticks sedan programstart
clock_t clock()

#define CLOCKS_PER_SEC ...
```

<chrono>, <ratio>

chrono: Tid i C++11

ratio: Bråk i C++11

<string>

size_t	size()
size_t	length()
string&	insert(index, tecken/sträng)
string&	erase(index, [antal])
char	at(index)
char*	c_str()
char	operator[index]
string&	operator+=(tecken/sträng)
string	operator+(tecken/sträng)
string	substr(index, antal)

<random>

Skapa motor:

```
default_random_engine def(seed);
mt19937 mt(seed);
random_device rnd;
```

Skapa distribution:

```
uniform_int_distribution<int> uni(start, stop);
normal_distribution norm(mean, stddev);
```

Slumpa:

```
uni(rnd); // slumpar mellan start och stop från rnd
norm(def); // slumpar normafördelat från def-motorn
```

`srand(), rand() i C-biblioteket`

<boost/regex.hpp>

sudo apt-get install libboost-all-dev

boost::regex(pattern)

boost::regex_match(subject, pattern)

boost::regex_search(subject, result, pattern)

boost::regex_replace(subject, pattern, replace)

C-sträng, C-array

```
int array[5] = {1, 2, 3, 4, 5};

const char* cstring = ”Kalle”; // vanligen
char const* cstring = ”Kalle”; // jag?
// egentligen:
const char data[] = {’K’, ’a’, ’l’, ’l’, ’e’, ’\0’};
const char* cstring = data;

// Rita! Förklara!
// Storlek? Lagras inte någonstans (om inte du gör det!)
```

Kommandoradsargument

```
int main(int argc, char* argv[])
{
  vector<string> arg(argv, argv+argc);

  for (unsigned i = 0; i < arg.size(); ++i)
  {
    cout << arg[i] << endl;
  }
  return 0;
}
```

Templates

- En mall för hur något skall utföras
- Vad det utförs på (datatyp) anges först vid användning
- Tills dess används platshållare
- Mallen kan ställa krav på vilka operationer datatypen måste klara av

Template: exempel

```
template<typename PLATSHÅLLARE>
void swap(PLATSHÅLLARE a, PLATSHÅLLARE b)
{
  PLATSHÅLLARE save = a; // kopiering
  a = b;                 // tilldelning
  b = save;
}
// swap() ställer kraven att datatypen
// PLATSHÅLLARE kan kopieras och tilldelas
```

Template: användning

```
int main()
{
  int a, b;
  float c, d;
  double e, f;
  swap(a, b);
  swap(c, d);
  swap(e, f);
  swap(a, f); //?
}
```

Template: kompilering

- Templatekod kan inte kompileras
- Platshållaren är okänd!
- Kan bara kompileras efter att platshållaren bestämts
- Undantag till reglerna om headerfiler och implementationsfiler
- Templateimplementationsfilen kompileras inte, den inkluderas sist i templateheaderfilen