Objektorienterad Programmering 8p
TDP004

Per-Magnus Olsson
perol @ida.liu.se
E-huset: 2F448

Introduktion

- Administrativ information.
- Föreläsningsplanering.
- Kort genomgång av grundläggande C++.

Administrativ information

- 8 föreläsningar inkl en repetitionsföreläsning.
- Laborationsdel.
- Hemarbete kommer att krävas.
- Alla laborationer i era egna datasalar.

Administrativ information, forts.

- Varje labbgrupp har tillgång till båda datasalarna.
- Föreläsningsbilderna finns på nätet.
- Föreläsningarna går igenom exempel, inte allt som ingår i kursen. Bok krävs.
- Räkna inte med att hinna göra allt på laborationerna.

Examinationsmoment

- Dugga U/G
  - Godkänt resultat ger poäng tillgodo på tentamen.
- Laborationsdel
- Tentamen.

Laborationsdel

- 3 st seminarier med obligatorisk närvaro.
  - Utblickar och diskussion.
- 3 st dojos med obligatorisk närvaro.
  - Diskussion och implementation av uppgifter i mindre grupper.
- 5 parvisa laborationer som lämnas in.
  - Obligatoriskt att använda Eclipse för laborationerna.
- Laborationsdelen ger 4 hp. Betyg U,G.

Tentamen

- Enskild datortentamen,. Betyg U, 3, 4, 5. Ger 4 hp.
- Man kan få poäng tillgodoräknade på tentamen genom att:
  - Klara duggan.
  - Vara närvarande på minst 6 av 8 föreläsningar.
- För godkänd kurs krävs båda momenten godkända och betyget på datortentamen blir kursbetyget.
- OBS! I studiehandboken heter labdelen LAB2 och tentan TEN2. Namnet LAB2 har inget med laboration 2 i labdelen att göra.

Kurslitteratur

- Rekommenderas starkt:
  - C++ Primer, 4th Edition. Lippman, Lajoie, Moo. Addison-Wesley.
- Möjlig:
  - C++ direkt, 2:a utgåvan. Skansholm. Studentlitteratur.
- Båda fungerar, C++ Primer är lite mera omfattande och lämplig att behålla efter kursen.

Föreläsningsplanering

- Fö 1 Administration, introduktion och grunder.
- Fö 2 Objektorientering grunder.
- Fö 3 Standardbiblioteket (STL).
- Fö 4 Strömmar. Utvärdering.
- Fö 5 Minneshantering.
- Fö 6 Objektorientering, arv, polymorfi.
- Fö 7 OO design, tips och råd.
- Fö 8 Repetition, frågestund.

C++

- Baserat på C.
- Möjlighet till objektorienterad programmering med klasser, polymorfi, operatoröverlagring mm.
- C++ är ett stort och flexibelt språk. Möjligt att programmera på både hög och låg nivå.

Ett första C++-program

```
#include<iostream>                                          // För att kunna mata in och ut.

int max(int x, int y)
{
    if(x > y)
    {
        return x;
    }
    else
    {
        return y;
    }
}

 int main()                                                 //Huvudprogram
{
    int i;
    int j;
    std::cout << "Input two integers: ";                   // Skriv ut instruktioner.
    std::cin >> i >> j;                                    // Spara de inmatade värdena.
    int greatest = max(i, j);                              // Anropa funktionen max.
    std::cout << "The greatest integer was "<< greatest <<std::endl;   // Skriv ut resultatet.
    return 0;                                              // Returnera
}
```

Programstruktur

- Programmet kan och bör delas upp i flera filer.
  - I headerfilen <filnamn>.h /.hpp definieras funktioner, datatyper mm.
  - I implementationsfilen <filnamn>.cc / .cpp implementeras funktionerna som definerades i headerfilen.
  - Logiskt med uppdelning vid OO-programmering.
- En main-funktion måste alltid finnas.
- Platt struktur, ej nästlade funktioner.

Datatyper

- Samlingsnamn för variabler och konstanter.
- Deklareras med <typ> namn.
- Tilldelning av värde vid deklaration kallas initiering.
- Ex:

```
int age;                        // heltalsvariabel
float salary = 1.2345;          // flyttalsvariabel som initieras
const double pi = 3.14159;      // flyttalskonstant, måste initieras
```

Datatyper, forts.

- Inbyggda datatyper:
  - Short, **int**, long – heltal.
  - float, **double**, long double – decimaltal.
  - bool – Boolesk variabel. Endast två giltiga värden: true, false.
  - **char**, wchar_t – bokstäver.
  - void - speciell typ, betyder ”inget värde”.
- Programmeraren kan definiera typer.

Datatyper, forts.

- En variabel kan tilldelas värde vid deklaration, sk initiering. En konstant måste initieras, och kan sedan inte ändras.

```
int units = 1,
int number(3);          //Inte vanligt men tillåtet. Samma som int number  = 3;
const int moreUnits = units + 2;
bool married = false;
char middleInitial = 'G';
```

Datatyper, forts.

- Olika typer för heltal skiljer sig åt pga vilka värden de kan spara.
  Ex: int kan innehålla större (och mindre) värden än short kan.
- Olika typer av flyttal skiljer sig åt pga vilken precision be har.
  Ex: double har större noggranhet än float.
- Bool representeras som ett heltal där 0 representerar false, *och alla andra värden true.*
  Ex: bool a = …
  cout<<a;
  Ger kanske värdet 12, dvs true.

Variabelscope

- ”Var en variabel finns tillgänglig”.
- En global variabel finns tillgänglig överallt.
- En lokal variabel finns så länge dess scope existerar.
- Ex:

Aritmetiska operatorer

- + - / * %
- % är modulo-operatorn. Återstoden vid heltalsdivision. Ex: 21 % 6 = 3
- Ex: op någon av ovanstående:

```
i <op>= j;    // Betyder i = i <op> j.
i += j;       // Betyder i = i + j;
```

Operatorer och operander

- Exempel på kombinationer av uttryck:

```
x = 3.25              // Tilldelning.
2 * pi * radius       // Multiplikation, sammansatt uttryck.
- x                   // Negation.
sin(x)                // Funktionsanrop.
x < y                 // Relationsuttryck.
```

Prioritet vid sammansatta uttryck

- Alla operatorer har en prioritet. En operator med högre prioritet har företräde jämfört med en operator med lägre prioritet.
  - Ex: a + b * c beräknas som a + (b * c).
- Om operatorerna har samma prioritet gäller en viss beräkningsriktning beroende på nivå:
  - Ex: a + b – c beräknas som (a + b) – c.
- Parenteser kan användas för att få en viss ordning:
  - Ex: i uttrycket (a + b) * c beräknas först (a + b).

Stegningsoperatorer

- Följande betyder samma sak:

  x = x +1;

  x += 1;

  x++;

  x++ (--) är den sk postfix stegningsoperatorn, vilket ökar (minskar) värdet med ett.

- Prefix stegningsoperatorn: ++x, --x.

  Skillnad mellan x++, ++x: se C++ Primer s. 162.

Operatorer forts.

- == test för likhet.
- != skiljt från
- >, >=, <=, <
- Ex:
  - bool a = true;
  - bool b = function();
  - bool isDifferent = a != b;
- Observera den synbara likheten mellan tilldelning = och jämförelse == !

Logiska operatorer

- bool, som tidigare två värden: true/false.
- && ”och” mellan två bool.
- || ”eller” mellan två bool.
- ! ”skiljt från” mellan två bool.
- Ex:
  - bool a = true, b = false;
  - bool c = a || b; // c = true
  - bool d = a && b; // d = false
  - bool e = a ! b; // e = true

Val- och upprepnings-satser

- if()-else.
- for(), while(), do-while().
- if()-else används vid val.
- for används mest för då man vet hur många iterationer som ska göras, annars används while eller do-while.
- Med kommandot break avbryter man switch, while, do-while, for-loop och hoppar till satsen närmast efter.

if-satsen

```
if(villkor)
{
    //Satser som utförs om villkor är sant.
}
else
{
    //Satser som utförs om villkor är falskt.
}
```

If-satsen, forts.

- Else-delen kan utelämnas.
  - Om detta görs sker ingenting sker om villkoret är falskt.
- If-satser kan nästlas.
- Vad händer om man har många olika alternativ, och ska göra olika saker beroende på något variabelvärde?

switch-satsen

```
switch(variabel)
    case 1:
    {
        // Satser som utförs om variabel har värde 1.
        // Observera avsaknaden av break;
    }
    case 2:
    {
        // Satser som utförs om variabel har värde 2.
        break;
    }
    …
    default:
    {
        //satser om utförs om variabel inte har något av ovanstående värden.
    }
```

switch-satsen, forts.

- Bra om det finns många olika värden.
- Observera break!
  - Ointuitiv språkkonstruktion!
- default är inte obligatoriskt, men uppmuntras.
- Variabeldeklarationer kan enbart ske i den sista case: samt i default.

for-satsen

```
for(initieringssats; villkorssats; förändringssats)
{
   //satser som utförs så länge villkorssatsen är uppfylld.
}
```

- Ex:

```
int i;
for(i = 0; i < 3; i++)
{
    std::cout<<i<<" ";
}
->
0 1 2.
```

for-satsen, forts.

- Beräkningsordning:
  1. Initieringssatsen utförs.
  2. Avbrottsvillkoret beräknas. Om detta är falskt avslutas for-satsen.
  3. Satserna ”i kroppen” utförs.
  4. Förändringssatsen utförs.

  Åter till punkt 2 ovan.

while-satsen

```
while(villkor)
{
        /* satser som utförs så
         länge villkor är uppfyllt */
}
```

- Om villkor är falskt, händer inget.
- Programmeraren måste själv avbryta mha tex break eller genom att se till att villkor evalueras till false.

do-while-satsen

```
do
{
   /* satser som utförs första gången
   och så länge villkor är uppfyllt */
}
while(villkor);        //Observera att do-while avslutas med;
```

- Genomlöps alltid minst en gång.
- Programmeraren måste själv avbryta mha tex break.

Typkonvertering

float h = 3.996;
int i = h;
i = ?

- i = 3 pga trunkering. Hade du tur fick du en varning av kompilatorn…!
- Implicit typkonvertering utan någon kontroll från användarens sida.
- Om vi vill ha kontroll av konverteringen används cast.

Typkonvertering, forts.

- I C görs typkonvertering enligt följande:

float h = 3.996;
int i = (int)h;
Vi ska **inte** använda konvertering som i C.

- I C++ används static_cast.

float h = 3.996;
int i = static_cast<int>(h); //Värdet på h ändras inte.

- static_cast syntax:
  static_cast<typ som vi vill konvertera till>(variabel)

Typkonvertering, forts.

- Det finns flera olika sorters cast:
  static_cast
  dynamic_cast
  reinterpret-cast
  const_cast
- Med const_cast kan vi konvertera från konstanter.
- Använd typkonverteringar med försiktighet.

Tecken och texter

- Inbyggda datatypen char.
- Datatypen string lagrar text.
- Vi återkommer till datatypen string i föreläsningen om STL.

Teckenhanteringsfunktioner

- En del användbara funktioner finns i <cctype>.
- char c = 'a';
- char b = toupper(c); // b = 'A'
- islower(c); // returnerar true.

Grundläggande om funktioner

- Funktion är ett viktigt koncept inom OO pga att den kan användas för inkapsling av data och beräkningar.
- En funktion ska utföra en eller ett fåtal saker och inte lämna oönskade sidoeffekter efter sig.
- Funktioner *deklareras* i h-filen:

returvärde funktionsnamn(parameterlista);

Ex:

float smallest(float valueA, float valueB);

är en funktion som heter smallest, returnerar en float, och tar två inparametrar av typen float: valueA och valueB.

Funktioner, forts.

- Ex:

```
float smallest(float valueA, float valueB)
{
    if(valueA < valueB)
    {
      return valueA;          // Returnerar resultat.
    }
    else
    {
      return valueB;          // Returnerar resultat.
    }
}
```

Returvärde och lokala variabler

- Antingen returnerar funktionen något eller så gör den inte det (returtyp void).
- Om något ska returneras görs detta med return variabel.
- Även om funktionen har returtyp void så kan man returnera från funktionen med return.
- Returtypen kan vara av godtycklig typ: inbyggd eller egendefinierad.
- Inuti funktionen kan variabler deklareras på vanligt sätt, dessa kallas lokala variabler.

Inparametrar

- Kallas även (in)argument.
- En funktion tar noll eller flera parametrar dvs data som den behöver för att utföra sin beräkning.
- När funktionen anropas måste parametrarna anges i samma ordning som de finns i funktionsdeklarationen.

Inparametrar, forts.

- Värdekopiering – call by value
  int max(int a,int b)
  a och b kopieras vid funktionsanrop.
  Ev. ändringar på a och b görs på kopiorna.
- Referensöverföring – call by reference
  int max(int& a,int& b)
  a och b refereras direkt, inga kopior skapas.
  Ev. ändringar på a och b görs på ”originalen”.
- Generellt vill vi undvika onödig kopiering av data.

const inparametrar

- Vi kan deklarera vanliga konstanter med ordet const.
- På samma sätt kan vi göra inparametrar konstanta.

int max(const int& a, const int& b);

- Värdet på parametrarna a och b kan nu inte ändras i funktionen max.

const returvärde

- Även en funktions returvärde kan deklareras const:

const int max(const int& a, const& int b);

- Den anropande funktionen kan då inte ändra på returvärdet.
- Exempel: du vill inte att nån ska ändra ditt personnummer.
- Man kan ha en egen kurs om hur man använder const…

Exempel

- Vi kan använda & för att slippa kopiera data i onödan. För att garantera att vi inte ändrar på datan const-deklarerar vi inparametern.
- Ex:
  int findGreatest(const std::list<int>& values);
- list är en lista som innehåller element av typen int, hör till standardbiblioteket (STL).
- Mha const & har vi åstadkommit säker och snabb kod. Kompilatorn hjälper oss att inte ändra på inparametern.

Default-parametrar

- int f(int a, int b, int c = 0); // i .h-filen
- Parameter c har ett default-värde, detta möjliggör följande anrop:
  int x = 2, y = 3, z = 4;
  f(x, y); men också f(x, y, z);
- Vi *kan* undvika att ge ett värde på parametern c då vi anropar f.
- Default-parametrar måste stå sist i parameterlistan.

Sammanfattning

- C++ ger möjlighet till OO.
- Datatyper.
- Variabler och konstanter.
- Repetitionssatser.
- Funktioner.

- Nästa föreläsning: grundläggande OO.