Kurs i C++

Pedher Johansson (pedher@cs.umu.se)
Thomas Johansson (thomasj@cs.umu.se)
Mattias Åsander (mattiasa@cs.umu.se)

Mål med kursen

- Fräscha upp programmeringskunskaper
- Skillnader mot andra språk (Java, C, …)
- Förstå kod
- Objektsorientering
- Bygga bibliotek

Begrepp inom programmering

Ett första exempel

Ett första exempel

```
int main()
{
    int month = 10;
    int days;

    if (month == 4 || month == 6
        || month == 9 || month == 11)
        days = 30;
    else if (month == 2)
        days = 28;
    else if (month >= 1 && month <= 12)
        days = 31;
    else
        days = -1;

    return 0;
}
```


Ett första exempel

- Main-funktionen
- Byggblock
- Variabler och Datatyper
- Villkorssatser och logiska uttryck

Main-funktionen

- Anropas vid programstart
- Returnerar ett heltalsvärde till OS
  - 0 om det går bra
  - positivt värde vid fel

Uttryck, satser

- Uttryck
  - Kombination av operatorer och operander (Artimetiska uttryck, tilldelningar, funktionsarnrop)
- Sats
  - Ett självständigt uttryck (Avslutas med ;)
- Kodblock
  - En samling satser (Omges vanligen med { })

Variabler och identifierare

- Namngiven referens till värde
- **Tilldelas** vänster till höger med =
- **Identifierare** skapas med tecken a-z, A-Z, 0-9, _. Inleds ej med siffra
- Kan ej vara samma som **reserverade ord** (int, if, return, ...)

Datatyper

- Heltal (char, short, **int**, long)
- Flyttal (float, **double)**
- Tecken (char)
- Boolska värden (bool)

Datatyper

	Antal byte	Minsta värde	Största värde
char	1-128		
signed **char**	1	-128	127
unsigned char	1	0	255
signed **int**	4	-2.147.483.648	2.147.483.647
unsigned int	4	0	4.294.967.295
signed **short** *int*	2	-32.768	-32.767
unsigned short *int*	2	0	65.535
signed **long** *int*	8	-2.147.483.648	-2.147.483.647
unsigned long *int*	8	0	4.294.967.295
float	4	±3,4E-38	±3,4E+38
double	8	± 1,7E-308	±1,7E+308
bool	1	**false**	**true**

Villkorssatser

- Villkorar ett kodblock med hjälp av ett villkorsuttryck

```
if (<villkorssuttryck>)
    <sats>
```

```
if (<villkorssuttryck>)
{
    <samling satser>
}
```

```
if (<villkorssuttryck>)
    <sats>
else
    <sats>
```

```
if (<villkorssuttryck>)
{
    <samling satser>
}
else
{
    <samling satser>
}
```

Villkorssuttyck

- Ska resultera i ett sant eller falsk värde (true/false eller 0/nollskillt)
- Kan byggas upp av logiska uttryck

	Operator
jämförelse	==, <, >, <=, >=, !=
eller	||
och	&&
icke	!

Prioritetsordning

- Operatorers prioritet styr **evalueringsordning** i ett uttryck
- Kan styras med ( )

Prio	Operator	Beskrivning	Prio	Operator	Beskrivning
2	foo() ++ --	funktionsanrop ökning av värde (postfix) minskning - " -	13	&&	logiskt och
3	!	logisk icke	14	||	logiskt eller
5	* / %	mult, div., rest	15	=, +=, -=, *=	tilldelning
6	+ -	add. subtr.	17	,	komma
8	< > <= >=				
9	== !=				

Exemplet - med switch

```
int main()
{
    int month = 10;
    int days;

    switch (month)
    {
    case 4: case 6: case 9: case 11:
        days = 30;
        break;
    case 2:
        days = 28;
        break;
    case 1: case 3: case 5: case 7: case 8: case 10:
        days = 31;
        break;
    default:
        days = -1;
    }

    return 0;
}
```


Ett andra exempel

Ett andra exempel

```
#include <iostream>
using namespace std;

int main(int argc, char* argv)
{
    int value;
    bool valid;

    do
    {
        cout << "Ange en siffra: "
        cin >> value;
        valid = value > 0;
        if (valid)
            cout << "Du angav siffran " << value << endl;
    }
    while (valid);

    return 0;
}
```


Ett andra exempel

- Inkludera bibliotek
- Namnrymder
- Enkel I/O
- Upprepningssatser

Inkludera bibliotek

- Ange bibliotek som ska användas med #include
- Ange ej .h som i C

Namnrymder

- Namnrymder används för att undvika namnkonflikter
- ::-operatorn används för att använda en variabel eller funktion i en specifik namnrymnd
- using namespace används för att använda en viss namnrymnd

```
#include <iostream>
...
std::cout << ”Hello” << std::endl;
```

```
#include <iostream>
using namespace std;
...
cout << ”Hello” << endl;
```



```
#include <iostream>
using namespace std;

namespace first
{
  int x = 30;
  int y = 40;
}

namespace second
{
  int x = 5;
  int y = 6;
}

int main (int argc, char** argv)
{
   using namespace first;

   // Ger utskrift 35
   cout << x + second::x << endl;

   // Ger utskrift 46
   std::cout << y + second::y << std::endl;

   return 0;
}
```


I/O med iostream

- I/O används för formaterad utskrift och inläsning
- Används tillsammans med operatorerna << och >>
- Mer detaljer senare i kursen

Utskrifter

- **cout** används för utskrift till stdout
- Används med operatorn <<
- konstanten **endl** används för nyradstecken
  - '\n' existerar fortfarande

```
cout << ”Hello world” << endl;
```


Upprepningssatser

- Upprepar ett kodblock så länge som ett villkorsuttryck är sant
- while

```
while (<villkorsuttryck>)
{
    <samling satser>
}
```

- do-while

```
do
{
    <samling satser>
}
while (<villkorsuttryck>);
```

Upprepningssatser - for

- for

```
for (<initialt uttryck>;
     <villkorsuttryck>;
     <uttryck innan upprepning>)
{
    <samling satser>
}
```

- Kan skrivas om till en while-sats

```
<initialt uttryck>;
while (<villkorsuttryck>)
{
    <samling satser>
    <uttryck innan upprepning>
}
```


Exempel på for

```
#include <iostream>
using namespace std;

int main()
{
    for (int i = 0; i < 10; i++)
    {
        cout << "Hello nr. " << i << endl;
    }

    int i = 0;              // initialt uttryck
    while (i < 10)          // villkorsuttryck
    {
        cout << "Hello nr. " << i << endl;
        i++;                // uttryck innan upprepning
    }
}
```


Ett tredje exempel

Ett tredje exempel

```
#include <cmath>
#include <iostream>
using namespace std;

double hypotenuse(double a, double b)
{
   return sqrt(pow(a,2) + pow(b,2));
}

int main()
{
   double a, b;

   cout << "Give the two sides: ";
   cin >> a >> b;

   cout << "The hypotenuse is " << hypotenuse(a, b) << endl;

   return 0;
}
```


Ett tredje exempel

- Anropa funktioner
  - argument
- Egna funktioner
  - parameter
  - returvärden
- Räckvidd av variabler

Anropa funktioner

- Givet noll eller flera **argument** inom parentes

```
int pow = 2;

pow(3, pow);
```

resulterar anropet (ev.) i ett **värde**

```
int res = pow (2,4);
```

Egna funktioner

- Givet noll eller flera **parameter**värden utförs en samling satser som resulterar i ett **returvärde**

Egna funktioner

- Funktions**deklaration** görs innan den används
  - **Namn**
  - **Returtyp**
    datatyp på returvärdet
  - **Parameterlista**
    namn och datatyp på samtliga parametrar
  - **Funktionskropp**
    samling satser samt en return-sats

Funktionsdeklaration

```
<returtyp> <namn>(<parameterlista>)
{
    <samling satser>
    <return-sats>
}

double hypotenuse(double a, double b)
{
    double value = sqrt(pow(a,2) + pow(b,2));
    return value
}
```


Anropa en egen funktion

- **Antal och datatyp** på **argumenten** måste stämma med **parametrarna**

```
double hypotenuse(double a, double b)
{
   return sqrt(pow(a,2) + pow(b,2));
}

int main()
{
   ...

   double value = hypotenuse(5.0, 3.2);

   ...

   return 0;
}
```


Räckvidd av variabler

- Variabel deklarerad i ett kodblock (**lokal variabel**) existerar enbart i blocket
- Variabel utanför något kodblock är en **global variabel**
- Vid namnkonflikt gäller den lokala variabeln
- Parametrar är lokala variabler


```
int main()
{
   if (true) {
      // a är lokal i kodblocket
      int a = 1;
   }

   // a finns inte i här
}
```

```
int foo()
{
   // a finns inte i foo()
   return 2;
}

int main()
{
   // a är lokal i main
   int a = 1;

   foo();
}
```

```
int foo()
{
   // b är en lokal variabel
   int b = 3
   return b;
}

int main()
{
   // b finns inte i main()
   foo();
}
```

```
// a är en global variabel
int a = 1;

int foo()
{
   // a finns både i foo()
   return 2;
}

int main()
{
   // ... och i main()
   foo();
}
```