TDT4102
Prosedyre- og objektorientert programmering, Øvingsforelesning veke 3, 18. januar

Sivert Krøvel
sivertkr@stud.ntnu.no

Øvingsforelesningane

- Vit.ass er ansvarlig, men fleire er involvert, deriblant undassar og stipendiat

Øvingsforelesningane

- Går igjennom relevant stoff for øvinga de jobbar med denne veka
- Praktiske eksempel og oppgåver
- Spørsmål og hjelp med øvingane

Agenda

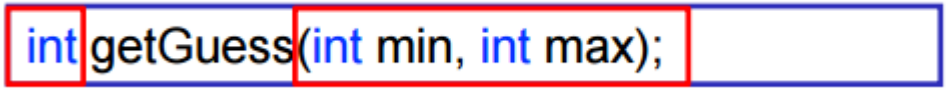
- Basics, litt repetisjon
- Kort oppgåve
- Skrive ut vs returnere og input
- Teste funksjonar
- Kort oppgåve
- Kva bør eg tenke på?
- Lengre oppgåver

Kva utgjer eit program?

```
1    #include<iostream>
2
3    int main(){
4        std::cout << "Hello World!" << std::endl;
5
6        return 0;
7    }
8
```

- Nøyaktig éin main-funksjon
- Definerer start- og sluttpunkt

Funksjonar



Lånt frå Lasse si forelesning 17/1

-Returtype

-Funksjonsnavn

-Parameterliste

Funksjonsprototypar

```
4   void printDate(int day, int month, int year);
5   bool isEven(int number);
6   double getArea(double radius);
7   int myFunc(int n);
```

- Gir grunnleggande informasjon om funksjonane dine
- Hjelpsomt med gode funksjons- og parameternavn!

Funksjonsdefinisjon

Sjølve funksjonskroppen

```
bool isEven(int number)
{
    if (number % 2 == 0)
        return false;
    else
        return true;
}
```

Å bruke funksjonar

- Ein funksjon må deklarerast før han kan brukast.
- Funksjonsprototypen må vere kjent for kallande funksjon (deklarert «over» i koden)
- Definisjonen kan godt komme seinare

Å bruke funksjonar

deklarasjon

```
bool isEven(int);
```

main

```
int main() {
    cout << isEven(2);
    return 0;
}
```

definisjon

```
bool isEven(int number)
{
    if (number % 2 == 0)
        return false;
    else
        return true;
}
```

Variablar

- Datatype og variabelnavn
- Må deklarerast
- Eksisterer kun i gyldigheitsområdet (scope) der den blei deklarert.
- Definert av blokker, {...}

```
14      {
15          int a = 1;
16          {
17              int b = a + 1;
18          }
19          a = a + b;
20      }
```

identifier "b" is undefined

Nokre datatypar

- int: heiltal
- double, float: flyttal (med desimalar)
- bool: true/false
- char: bokstavar
- (std::)string: tekstreng

Andre datatypar

- Det fins også meir «kompliserte» og eigendefinerte typar/klasser
  - std::ostream: output-strøm
  - sf::Window: vindusklasse frå tredjepartsbibliotek

Desse treng vi ikkje tenke så mykje på akkurat no, men dei kjem att seinare i pensum

Oppgåve:

Lag ein funksjonsprototype for ein funksjon som samanliknar to heiltal, og returnerer det minste av dei

(ca 1 min)

Oppgåve:

Lag ein funksjonsprototype for ein funksjon som samanliknar to heiltal, og returnerer det minste av dei

(ca 1 min)

```
int min(int a, int b);
```

Agenda

- Basics, litt repetisjon
- Kort oppgåve
- Skrive ut vs returnere og input
- Teste funksjonar
- Kort oppgåve
- Kva bør eg tenke på?
- Lengre oppgåver

Skrive ut til skjerm

- I matlab/python brukar vi funksjonar (print, disp, fprintf)
- I C++ brukar vi som regel ein output-strøm (ostream) og insertion-operatoren (<<). Vi kjem tilbake til det seinare.

```
std::cout << "Hello World!" << std::endl;
```

Skrive ut/printe vs returnere

Funksjonar som skriv ut

- Resultatet vert ikkje lagra, kun printa
- «Forsvinn» når funksjonen er ferdig
- Nyttig for å formatere output
- Ikkje så vanlig i øvingsopplegget

Funksjonar som returnerer

- Resultatet vert sendt vidare
- Tilgjengelig for andre funksjonar
- Nyttig når vi får bruk for resultatet seinare
- Vanlig i øvingsopplegget

Skrive ut/printe vs returnere

Funksjon som skriv ut

```
void isEven(int number)
{
    if (number % 2 == 0)
        cout << number << " is even\n";
    else
        cout << number << " is odd\n";
}
```

Funksjon som returnerer

```
bool isEven(int number)
{
    if (number % 2 == 0)
        return true;
    else
        return false;
}
```

Tenk over om du vil returnere eller skrive ut resultatet. Det er sjelden nødvendig å gjere begge deler.

Input frå brukar

- I matlab/python brukar vi funksjonar (input)
- I C++ brukar vi ein input-strøm (istream) og extraction-operatoren (>>)

```
int a, b, c;
std::cin >> a >> b >> c;
```

Hente input frå brukar vs ta inn som parameter

Input som parameter

- Nyttig når ein brukar verdiar frå andre funksjonar etc.
- Programmet stoppar ikkje opp undervegs
- Form er definert på førehand (datatype, antall parameter...)

Input frå brukar (tastatur)

- Nyttig som utgangspunkt
- Programmet stoppar opp og avventar svar frå konsollen
- Brukaren kan misforstå, udefinert input

Hente input frå brukar vs ta inn som parameter

Input som parameter

```
bool isPrime(int num) {
    //kode her...
}
```

Input frå brukar (tastatur)

```
bool isPrime() {
    int num;
    std::cout << "Skriv inn eit tal: ";
    std::cin >> num;
    //kode her
}
```

Agenda

- Basics, litt repetisjon
- Kort oppgåve
- Skrive ut vs returnere og input
- Teste funksjonar
- Kort oppgåve
- Kva bør eg tenke på?
- Lengre oppgåver

Teste funksjonar

- Det er mykje ein kan seie om å teste software, men...
- I dette emnet forventar vi enkel testing
- T.d. kjøre funksjonen i main med nokre inputverdiar du kjenner til resultatet for.
- Overbevise deg sjølv (og studass) om at funksjonen gjer det han skal

Teste funksjonar

- Ofte får du i oppgåve å lage ein funksjon som skal utføre ei bestemt oppgåve
- Nødvendig å *teste* om funksjonen utfører oppgåva riktig
- Ein feil kan lett forplante seg når prosjekta vert større; å teste funksjonar tidlig gjer det lettare å luke ut slike feil
- Enkelt oppsummert: Sjekk at funksjonen gir forventa resultat for kjende inputverdiar.

Demonstrasjon

Kvifor er det viktig å teste funksjonane undervegs?

Oppgåve

Test denne funksjonen i main (ca 3 min):

```
bool isEven(int number)
{
    if (number % 2 == 0)
        return false;
    else
        return true;
}
```

Løysings*forslag*

```
int main() {
    //test, isEven
    for (int i = -5; i < 6; i++) {
        cout << "isEven(" << i << "): "
            << isEven(i) << endl;
    }
}
```

```
isEven(-5): 1
isEven(-4): 0
isEven(-3): 1
isEven(-2): 0
isEven(-1): 1
isEven(0): 0
isEven(1): 1
isEven(2): 0
isEven(3): 1
isEven(4): 0
isEven(5): 1
Trykk en tast for å fortsette...
```

Agenda

- Basics, litt repetisjon
- Kort oppgåve
- Skrive ut vs returnere og input
- Teste funksjonar
- Kort oppgåve
- Kva bør eg tenke på?
- Lengre oppgåver

Kva bør eg tenke på når eg lagar ein funksjon?

- Kva er oppgåva til funksjonen?
  Søk hjelp i funksjonsnavnet
- Skal funksjonen returnere noko?
  Skal den printe noko?
- Kva treng den av input? Er det tilstrekkelig med parametrar, eller treng den input frå konsoll (cin)?
- Fungerer den som den skal? Korleis teste?

Døme: Øving 1 oppg 2f

Matlab

```
function [ primeness ] = isPrime( n )
    primeness = true;
    for i = 2:(n-1)
        if mod(n,i) == 0
            primeness = false;
            break;
        end
    end
end
```

Python

```
def isPrime(n):
    primeness = True
    for j in range(2,n):
        if n%j == 0:
            primeness = False
            break
    return primeness
```

Kva gjer funksjonen?
Skal vi returnere noko? Kva?
Skal vi ta inn parametrar? Kva type?

Oppgåver

Finn passande navn, returtypar og argument

1) Lag ein funksjon som tek inn to flyttal og returnerer det største av dei
2) Lag ein funksjon som tek inn eit heiltal og skriv ut talet utan forteikn (absoluttverdien)
3) Implementer funksjonane, og test dei i main.
4) (Bonus): Lag ein funksjon som skriv ut alle tall som er delelig med 3, opp til eit tal *limit*, gitt som parameter (hint: for-løkke, modulo-operatoren %)