Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4102 Prosedyre og Objektorientert programmering Vår 2015

Øving 1 LF

LØSNINGSFORSLAG

Mål for denne øvinga:

- lære grunnleggende C++ og prosedyreorientert programmering
- lære hvordan programmer lese inn data fra brukeren og skrive ut på skjermen
- lære hvordan funksjoner kan ta inn data som parametre og gi returverdier

Generelle krav:

- Bruk de eksakte navn og spesifikasjoner gitt i oppgaven.
- Det er valgfritt om du vil bruke en IDE (Visual Studio, XCode), men koden må være enkel å lese, kompilere og kjøre.

Anbefalt lesestoff:

- Kapittel 1, 2 og 3, Absolute C++ (Walter Savitch)
- Notater fra fagets side på It's Learning

1 Funksjoner og «Input/Output» (25%)

a) Skriv en funksjon som heter `inputAndPrintInteger` som lar brukeren skrive inn et heltall og skriver det ut til skjerm.

Løsningsforslag:

```
void inputAndPrintInteger() {
    int number = 0;
    cout << "Skriv inn et tall: ";
    cin >> number;
    cout << "Du skrev: " << number << endl;
}
```

b) Skriv en funksjon som heter `inputInteger` som lar brukeren skrive inn et heltall og *returnerer* dette.

Løsningsforslag:

```
int inputInteger() {
    int number = 0;
    cout << "Skriv inn et tall: ";
    cin >> number;
    return number;
}
```

c) Skriv en funksjon `inputIntegersAndPrintSum` som ved å bruke en av funksjonene du nå har skrevet, leser inn to heltall fra brukeren og skriver ut kun summen til skjermen.

Løsningsforslag:

```
void inputIntegersAndPrintSum() {
    int firstNumber = inputInteger();
    int secondNumber = inputInteger();
    cout << "Summen av tallene: " << firstNumber + secondNumber << endl;
}
```

d) Skriv en kommentar i koden din som forklarer hvilken funksjon av `inputAndPrintInteger` og `inputInteger` du brukte for å lage `inputIntegersAndPrintSum`, og hvorfor.

Løsningsforslag:

Her må vi bruke `inputInteger`. `inputAndPrintInteger` ville bare skrevet ut enkeltverdiene, uten å ta vare på dem på noen måte slik at vi kunne gjort noe mer med dem (som å summere dem, slik vi skal i denne oppgava). Dette demonstrerer dermed at det finnes tilfeller der vi trenger verdien som retur. Vi er heller ikke egentlig interessert i å skrive ut de verdiene vi nettopp skrev ut til skjerm ukritisk, vi vet tross alt hva vi nettopp skrev inn, og er mer interessert i å skrive ut summen av dem. Dermed vil verdiretur være riktig, og ikke verdiutskrift.

e) **Skriv en funksjon som heter `isOdd`, som tar inn et heltall og returnerer en en boolsk verdi. Funksjonen skal returnere `true` dersom argumentet er et oddetall og `false` ellers. Den skal *ikke* lese noe inn fra brukeren eller skrive ut noe til skjermen.**

Løsningsforslag:

```
bool isOdd(int n) {
    return n % 2;
}
```

Svaret over er så kort fordi det utnytter at sannhetsverdien *false* lagres som tallet 0, og alle andre tallverdier regnes som *true*. F.eks. gir 10 % 2 oss 0, som konverteres implisitt til *false*, mens 11 % 2 gir oss 1, som konverteres til *true*.

Under følger et alternativt løsningsforslag som i praksis gjør det samme som det første løsningsforslaget, men som kanskje kan være litt lettere å forstå:

```
bool isOdd(int number) {
    if (number % 2 == 1) {
        return true;
    } else {
        return false;
    }
}
```

f) Skriv en funksjon som heter `printHumanReadableTime` som tar inn (vha et parameter) antall sekunder, og skriver dette ut til skjerm i et menneskevennlig format. For eksempel vil 10000 sekunder skrives ut som "2 timer, 46 minutter og 40 sekunder".

Løsningsforslag:

```
void printHumanReadableTime(int seconds) {
    // Antall timer er sekunder / 3600
    // Siden begge operandene er heltall utføres
    // heltallsdivisjon. Desimaldelen forsvinner m.a.o.
    int hours = seconds / 3600;
    // La antall sekunder være resten fra divisjonen over.
    // Modulo (%) gir heltallsresten fra en divisjon, f.eks. har 10/3,
    // 3 som heltallssvar, og 1 som rest (10 = 3 * 3 + 1). Vi trenger
    // ikke lengre alle sekundene som har gått opp i hele timer.
    seconds = seconds % 3600;
    // Minuttene er da resterende sekunder / 60.
    int minutes = seconds / 60;
    // Og rene sekunder er resten fra denne divisjonen igjen
    seconds = seconds % 60;

    cout << hours << " time";
    if (hours != 1) {
        cout << "r";
    }
    cout << ", " << minutes << " minutt";
    if (minutes != 1) {
        cout << "er";
    }
    cout << " og " << seconds << " sekund";
    if (seconds != 1) {
        cout << "er";
    }
    cout << endl;

    /* Mer kompakt løsning som benytter ternary operator:
    cout << hours
         << " time" << (hours == 1 ? "" : "r") << ", ";
         << minutes
         << " minutt" << (minutes == 1 ? "" : "er") << ", og ";
         << seconds
         << " sekund" << (hours == 1 ? "" : "er") << endl;
     */
}
```

2 Løkker (15%)

a) Lag en funksjon som heter `inputIntegersUsingLoopAndPrintSum`, som tar utgangspunkt i koden i funksjonen `inputIntegersAndPrintSum` fra oppgave 1. Den nye funksjonen skal la brukeren velge hvor mange tall som skal summeres, enten ved å angi antallet tall først, eller ved å slutte når brukeren skriver inn tallet 0.

Løsningsforslag:

```
// 2a - For-løkke-løsning
void inputIntegersUsingForLoopAndPrintSum() {
    int nTerms = 0;
    cout << "Hvor mange tall vil du summere? ";
    cin >> nTerms;

    int sum = 0;
    for (int i = 0; i < nTerms; i++) {
        int term = inputInteger();
        sum += term;
    }
    cout << "Summen av tallene: " << sum << endl;
}

// 2a - While-løkke-løsning
void inputIntegersUsingWhileLoopAndPrintSum() {
    cout << "Skriv inn tall du vil summere. "
         << "Skriver du inn 0 avsluttes løkken og summen skrives ut."
         << endl;

    int sum = 0;
    int term = inputInteger();
    while (term != 0) {
        sum += term;
        term = inputInteger();
    }

    cout << "Summen av tallene: " << sum << endl;
}

// 2a - Do-While-løkke-løsning
void inputIntegersUsingDoWhileLoopAndPrintSum() {
    cout << "Skriv inn tall du vil summere. "
         << "Skriver du inn 0 avsluttes løkken og summen skrives ut."
         << endl;

    int sum = 0;
    int term = 0;
    do {
        term = inputInteger();
        sum += term;
    } while (term != 0);

    cout << "Summen av tallene: " << sum << endl;
}
```

b) Den forrige deloppgaven kan løses enten ved at brukeren angir antall tall som skal summeres i forkant, eller at man fortsetter å lese inn tall helt til brukeren skriver inn 0. **Skriv en kommentar i koden din som forklarer hvilken type løkke som er best egnet for hver av de to alternativene, og hvorfor.**

Løsningsforslag:

Hvis vi vet hvor mange ganger vi skal repetere noe når vi starter løkken, bruker vi typisk en for-løkke. I det første tilfellet i oppgave 2a har vi "angi antallet tall først" i oppgaveteksten, som betyr at antallet iterasjoner er kjent før vi begynner å iterere, ergo for-løkke. Hvis vi derimot ikke vet hvor mange ganger vi skal iterere, men heller avhenger av noe som skjer underveis for å finne antallet ganger, har vi typisk en while-løkke. While-løkke kommer som navnet hinter til av at vi skal gjøre noe "så lenge" noe er sant. I dette tilfellet skal leddet ikke være 0, og vi kan ikke vite når brukeren bestemmer seg for at "nok er nok".

Det er verdt å nevne at ei for-løkke også kan skrives som en while, og vice versa, så begge alternativene i 2a KAN teknisk sett løses med begge typer løkke. Det må dog nevnes at det blir litt mer kronglete, da for-løkker ikke egentlig egner seg veldig godt til tilfellet hvor antallet iterasjoner er helt ukjent, og while-løkker har ikke tellevariabler med i grunnlaget, slik at det blir litt merarbeid om man skal skrive ei telleløkke som while-løkke.

Det skal også nevnes at do-while-løkke kanskje er enda mer passende enn while-løkke for det alternativet, da vi alltid vil at brukeren skal bli spurt om input minst en gang.

c) **Skriv funksjonen `inputDouble` som skal gjøre det samme som `inputInteger` fra oppgave 1, men istedenfor å lese inn et heltall, skal denne funksjonen lese inn og returnere et desimaltall.**

Løsningsforslag:

```
double inputDouble() {
    double number = 0;
    cout << "Skriv inn et tall: ";
    cin >> number;
    return number;
}
```

d) Skriv en funksjon som konverterer NOK til Euro.

Løsningsforslag:

```
void convertNOKtoEUR() {
    double NOK = -1.0f;
    while (NOK < 0.0f) {
        NOK = inputDouble();
    }

    double EUR = NOK / 9.12;

    cout << setprecision(2) << fixed
         << NOK << " NOK er "
         << EUR << " EUR" << endl;

    /** "Magic formula" page 31 in textbook - As an alternative to << setprecision(2
      cout.setf(ios::fixed);
      cout.setf(ios::showpoint);
      cout.precision(2);
    */
}
```

e) Skriv en kommentar i koden din som forklarer hvorfor vi ikke bør bruke `inputInteger` i forrige oppgave, men heller `inputDouble`, legg spesielt merke til hva i oppgaveteksten som legger føring på denne bruken. Kommenter også på returtypen til funksjonen du skrev i forrige oppgave.

Løsningsforslag:

`inputInteger` gir oss et heltall, dette vil dermed ikke ha noen desimalplasser, som vil være problematisk når vi skal regne med valuta. Rett nok vil resultatet fortsatt bli regnet ut med riktig kurs, men øre-biten av verdien vil bli borte allerede FØR utregningen, noe som vil gi oss et svar som neppe er helt riktig. Når vi skal regne med desimaltall, bør vi derfor være nøye med å velge riktig datatype. I dette tilfellet er double tilstrekkelig, selv om double ikke er eksakt. Vi beholder dermed desimalplassene, og får et mer generelt korrekt svar fra utregningen. Det i oppgaven som legger føring på valg av datatype i dette tilfellet er ordleggingen "to desimaler", som gir oss et hint om at det her skal jobbes med desimaltall.

Returtypen til funksjonen er void, siden oppgaven sier "skriv ut det vekslede beløpet", og ikke "returner det vekslede beløpet".

3 Menysystem (10%)

a) Skriv om `main` slik at brukeren kan velge i en meny mellom funksjonene fra foregående oppgaver.

Eksempel på menysystem med hjelpefunksjoner:

```
void testIsOdd() {
    int number = inputInteger();
    if (isOdd(number)) {
        cout << number << " er et oddetall." << endl;
    } else {
        cout << number << " er et partall." << endl;
    }
}

void testPrintHumanReadableTime() {
    int nSeconds = inputInteger();
    printHumanReadableTime(nSeconds);
}

int main(int argc, char **argv) {
    int valg = -1;
    while (valg != 0) {
        cout    << "0) Avslutt" << endl
                << "1) Summer to tall" << endl
                << "2) Sjekk om et tall er et oddetall" << endl
                << "3) Konverter sekunder" << endl
                << "4) Summer flere tall (for)" << endl
        cout << "Angi valg (0-4)";
        cin >> valg;
        cout << endl;
        switch (valg) {
        case 0:
            break;
        case 1:
            inputIntegersAndPrintSum();
            break;
        case 2:
            testIsOdd();
            break;
        case 3:
            testPrintHumanReadableTime();
            break;
        case 4:
            inputIntegersUsingForLoopAndPrintSum();
            break;
        default:
            cout << "Ugyldig valg" << endl;
        }
        cout << endl;
    }
}
```

b) Skriv en funksjon som skriver ut en gangetabell på skjermen (cout). La brukeren gi både bredde og høyde på tabellen.

Løsningsforslag:

```
void printMultiplicationTable() {
    int height = 0;
    int width = 0;
    cout << "Hoyde?";
    cin >> height;
    cout << "Bredde?";
    cin >> width;
    for (int x = 1; x <= width; x++) {
        for (int y = 1; y <= height; y++) {
            cout << x *y << "\t";
        }
        cout << endl;
    }
}
```

4 Bruk av funksjoner i funksjoner, og røtter (25%)

I funksjonene under skal du tolke hva som skal være argumenter til funksjonen, og lære å bruke funksjoner i funksjoner. Generelt heretter i faget, skal funksjoner returnere verdier, og ikke skrive ut noe til skjerm, med mindre annet er spesifisert.

a) Skriv en funksjon `discriminant` som regner ut

$$b^2 - 4ac$$

og returnerer verdien (ingen utskrift til skjerm)

Løsningsforslag:

```
double discriminant(double a, double b, double c) {
    return pow(b, 2.0) - 4 * a * c;
}
```

b) Skriv en funksjon `printRealRoots` som finner de reelle røttene til andregradsligningen

$$ax^2 + bx + c = 0$$

ved å bruke formelen

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

der a, b og c er gitt som argumenter til funksjonen. Gjenbruk funksjonene fra de forrige deloppgavene, og skriv ut løsningene til skjerm.

Løsningsforslag:

```
void printRealRoots(double a, double b, double c) {
    int d = discriminant(a, b, c);
    if (d > 0.0) {
        double firstSolution = (-b + sqrt(d)) / (2 * a);
        double secondSolution = (-b - sqrt(d)) / (2 * a);
        cout << "Løsning 1: " << firstSolution << endl;
        cout << "Løsning 2: " << secondSolution << endl;
    } else if (d == 0.0) {
        double solution = -b / (2 * a);
        cout << "Løsning: " << solution << endl;
    } else /* (d < 0.0) */ {
        cout << "Ingen reell løsning finnes." << endl;
    }
}
```

c) Lag en funksjon `solveQuadraticEquation` som lar brukeren skrive inn 3 desimaltall og bruk `printRealRoots` til å regne ut røttene til andregradsuttrykket gitt ved disse tallene

Løsningsforslag:

```
void solveQuadraticEquations() {
    double a = 0.0;
    double b = 0.0;
    double c = 0.0;
    cout << "A: ";
    cin >> a;
    cout << "B: ";
    cin >> b;
    cout << "C: ";
    cin >> c;
    printRealRoots(a, b, c);
}
```

d) Legg til `solveQuadraticEquation` i testmenyen i `main()`

Se oppgave 3a for løsningsforslag.

e) Bruk testmenyen til å finne røttene til disse andegradsligningene. Sjekk at programmet fungerer med tall som gir 0, 1, og 2 løsninger

printRealRoots(1, 2, 4):

- Ingen reell løsning finnes.

printRealRoots(4, 4, 1):

- Løsning: -0.5

printRealRoots(8, 4, -1):

- Løsning 1: 0.183013
- Løsning 2: -0.683013

5 Flere Løkker (25%)

a) Skriv en `calculateLoanPayments` funksjon som regner ut årlige innbetalinger av et lån over 10 år. La brukeren spesifisere lånebeløp og rente. La programmet for hver innbetaling skrive ut (cout)størrelsen på innbetalingen og det gjenstående lånet. Utskriften skal være i et lettleselig format.

b) Utvid funksjonen fra forrige deloppgave til å skrive ut oversikten over innbetalinger som en pen og pyntelig tabell

Løsningsforslag:

```
void calculateLoanPayments() {
    double amount, interest;
    double total = 0;
    cout << "Laanesum: ";
    cin >> amount;
    cout << "Laanerente: ";
    cin >> interest;
    cout << "\t\tBetaling\tRest\n";
    for (int i = 0; i < 10; i++) {
        int payment = amount / 10
                    + (10 - i) / 10.0 * amount * (interest / 100.0);
        total += payment;
        cout << "Aar nr " << i + 1 << ":\t"
             << payment << "\t\t"
             << (9 - i) / 10.0 * amount << endl;
    }
    cout << "Totalt:\t\t" << total << endl;
}
```