```
/**
 *  En KODEBLOKK er et segment i koden som tilhører if, else, for, while, do,
 *  switch, en funksjon, en struct eller en class.
 *
 *  Nedenfor er det konsekvent brukt fire blanke tegns innrykk
 *  (i mye av koden i emnet er det brukt to eller fire tegns innrykk).
 *
 *  Ha minst EN blank mellom if/for/while/switch og etterfølgende '('.
 *
 *  Ha minst EN blank (som konsekvent nedenfor) før startende '{' av en blokk.
 *
 *
 *  Fordelene ved konsekvent bruk av innrykk og blanke tegn:
 *    - Andre kan raskere sette seg inn i din kode
 *      (da koden er godt lesbar og oversiktlig).
 *    - Ser raskt/enkelt hvilke if'er og else'r som hører sammen,
 *      og hvor løkker stanser.
 *    - Man ser enklere hvilke variabler som kan brukes hvor,
 *      og hvor de hører til.
 *    - Raskere debugging, da det er enklere å orientere seg.
 *    - Enklere å sette seg inn i egen kode - som man skrev for en tid siden.
 *
 *
 *  Hvor langt til høyre selve kommentarer bør begynne, for helst å holde seg
 *  totalt innenfor 80 tegn, er noe smak og behag. Men det er ofte en fordel
 *  at det starter såpass til høyre at det for øyet er enkelt å lese selve
 *  koden separat, uten å bli forstyrret av stadig innsmettet kommentering.
 *  Så flytter man heller blikket litt til høyre, når det er ønskelig å få
 *  med selve kommentaren. Dvs. at kommentarer bør starte tidligst på høyde
 *  med programsetningene i den blokken de er en del av, dvs. slik:
 *        if (.....) {
 *             //  Kommentar .....
 *            int tall = 5;
 *            //  Kommentar .....
 *            while (tall < 10) {
 *                //  Kommentar .....
 *                cout << tall++ << ' ';
 *            }
 *            //  Kommentar .....
 *        }
 *  Emnelærer synes personlig at kommentarer gjerne kan begynne enda lengre
 *  mot høyre, f.eks. et sted mellom 30. og 40.kolonneposisjon.
 *  Det viktigste er uansett at man er konsekvent mot den regelen man velger!
 *  Synes man det blir igjen lite plass på linjen for selve kommentarer
 *  (før man når 80 tegn), er det jo både lov og mulig og legge kommentarer
 *  på den forutgående linjen(e), og da gjerne avslutte med ':'
 *  (som dermed henviser til koden på neste linje).
 *  Dette med 80 tegn har sitt utspring i at dette er ofte linjelengden på
 *  utskrifter, samt linjelengden når koden gjøres om til pdf.
 *
 *
 *  Linjer nedenfor med KUN '///' symboliserer en eller flere linjer med kode.
 *
 *
 *  @author  André Revå Marthinsen  & Frode Haug, NTNU
 */
```

```
void kodeBlokk() {
               //  Alle linjer med kode mellom kodeBlokk() { og avsluttende }
               //  er innmat av en blokk. Alle linjer inne i en blokk skal
               //  ha samme innrykk (med noen få unntak).
    if (.....) {
               //  Dette er en blokk inne i en blokk. Alt inne i denne blokken
               //  skal rykkes inn i forhold til den ytre kodeblokken.
        int tall = 5;
               //  En annen blokk med eget innrykk:
        while (tall < 10) {
            ///
        }

        do {    // Og enda en annen blokk:
            ///
        } while (tall < 100);

    }

    if (.....) //  Her er det IKKE { }, fordi innmaten er KUN på én kodelinje.
        ///

    if (.....) {
        if (.....) {
               //  Definering av noen lokale variabler inn i en blokk:
            int alder = 21;
            string setning = "abcdefg";
               //  Setninger kan fort gå over 80 tegn, spesielt når det er
               //  mye innrykk, og spesielt om hvert av dem er ganske lange:
            alder = lesInt("Hvor mange ganger skal denne løkken gå", MINLOOPING, MAXLOOPING);
            cout << "Utskrift av variablene:\n\tAlder: " << alder << "\n\tSetning: " << setning
               //  Da er det bedre å dele opp linjen der det føles naturlig:
            alder = lesInt("Hvor mange ganger skal denne løkken gå",
                           MINLOOPING, MAXLOOPING);
            cout << "Utskrift av variablene:\n\tAlder: " << alder
                 << "\n\tSetning: " << setning << '\n';
        } else {
            int tall = 0;
               //  Ha gjerne TO blanke mellom leddene inni en for-løkke:
            for (int i = 0;  i < 10;  i++) {
                ///
            }

            switch (tall) { //  Switch med flere kodelinjer pr.case.
                case 1: {
                    ///
                    if (.....) {
                        //
                    } else {
                        ///
                    }
                    ///
                } break;

                case 2: {
                    ///
                } break;
            }
        }
               //  En rekke med if-else kan fort bli uoversiktlig om
    } else {    //  man ikke er konsekvent med bruk av innrykk.
        ///
    }
}
```

```
/**
 *  To eksempler med struct og class:
 */
struct Struct {
    string navn,
           adresse;
    int   alder;
    float hoyde;
};

class Klasse  {
    private:

    public:
        Klasse() {    }
        ~Klasse() {    }
        void lesData() {
            ///
        }
        void skrivData() const;     //  Definisjonen er et annet sted i filen.
};


/*****************************************************************************
 *  Ett KONKRET eksempel:
 *
 *  Ukonsekvent/uoversiklig bruk av innrykk. Vanskelig å følge programflyten:
 */
void roteteIndentering() {
  int tall = 0;
for (int i = 0; i < 5; i++) {
    tall += i;
}
    if (tall > 5)
      tall *= 5;
     while (tall < 100)
    tall++;

if (tall > 100) {
tall = tall * tall;
  } else if (tall > 200) {
  tall = tall / 2;
}
else tall = 10;
}

/**
 *  Den samme koden, med konsekvent bruk av innrykk ifm. kodeblokker
 *  - langt enklere å følge:
 */
void ryddigIndentering() {
    int tall = 0;
    for (int i = 0; i < 5; i++) {
        tall += i;
    }
    if (tall > 5)
        tall *= 5;

    while (tall < 100)
        tall++;

    if (tall > 100) {
        tall = tall * tall;
    } else if (tall > 200) {
        tall = tall / 2;
    } else
        tall = 10;
}
```