Øving 3
TDT4102

Frist: 15.2.2013

Mål for denne øvingen:

- Lage og bruke tabeller (arrays) med forskjellige datatyper
- Jobbe med tabeller og funksjonen
- Lage et fullstendig program (et enkelt spill)

Generelle krav:

- bruk de eksakte navn og spesifikasjoner som er 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 5, Absolute C++ (Walter Savitch)

NB: Hele øvingen kan gjøres i ett Visual C++ prosjekt, men det anbefales å lage en fil for hver del for å gjøre koden mer oversiktelig.

Del 1: Tabeller (arrays) (15 poeng)

Tabeller er nyttige for å lagre forskjellige typer data, og kan brukes for å lagre lignende data i et sammenhengende minneområde. I den første deloppgaven, skal du lage og bruke en tabell med telefonnumre

a) **(7,5%)** Lag en ny fil med en main-funksjon. Lag en tabell *phoneNumbers* av typen *int* i main-funksjonen. Legg inn fem forskjellige telefonnumre i tabellen.

b) **(7,5%)** Skriv funksjonen *printArray* som inneholder en løkke som skriver ut alle numrene i tabellen du nettopp laget. Kall denne funksjonen fra main(). Funksjonen din skal være i stand til å skrive ut tabeller av enhver lengde.

Hint: Denne funksjonen skal ta to argumenter.

Del 2: Flerdimensjonale tabeller (25 poeng)

a) **(5%)** Lag en 2-dimensjonal tabell for å lagre hvor mange ganger hvert av telefonnumrene i tabellen fra del 1 blir ringt. La hver rad i tabellen tilsvare et telefonnummer og hver kolonne en ukedag. (tabell[1][2] skal dermed representere hvor mange ganger *phoneNumbers[1]* ble ringt på onsdag.)

b) **(5%)** Skriv funksjonen *printArray* som tar den 2-dimensjonale tabellen som argument og skriver den ut på skjermen i et passende format.

Hint: Dimensjonen til tabellen vil alltid være [# telefonnumre] x [# ukedager (7)]

c) **(7,5%)** Skriv funksjonen *randomizeList* for å fylle tabellen med tilfeldige telefonsamtaler. Funksjonen skal ta den 2-dimensjonale tabellen og et heltall (*antallOppringninger*) som argumenter, sammen med alle andre argumenter du anser for nødvendig. Funksjonen skal kjøre en løkke et antall ganger spesifisert av *antallOppringninger* og for hver gjennomkjøring inkrementere en tilfeldig tabellpost.

Denne funksjonen skal ikke returnere noe, men gjøre alle endringer på tabellen som gies som argument.

Hint: Når du gir en tabell som argument, vil den bli gitt til funksjonen på en måte lik «call-by-reference» og du kan derfor endre tabellen direkte, uten å ta andre skritt.

d) **(7,5%)** Skriv funksjonen *countCalls* som tar den 2-dimensjonale tabellen, samt et heltall som spesifiserer en ukedag, som argument. Funksjonen skal returnere det totale antall telefonsamtaler for gitt ukedag som ett heltall.

Del 3: Mastermind (60 poeng)

I denne deloppgaven skal du implementere spillet Mastermind. Programmet ditt skal lage en tilfeldig kode på 4 bokstaver (A-F). Brukeren av programmet skal deretter gjette hvilke bokstaver det er i koden og hvilken rekkefølge de er i. Etter hver gang brukeren har gjettet, skal programmet fortelle brukeren hvor mange bokstaver brukere gjetter riktig, og hvor mange bokstaver som er riktig plassert. Det er ikke nødvendig å gjøre nøyaktig som beskrevet under, så fremt funksjonaliteten blir den samme og det benyttes arrays og fornuftig valgte funksjoner.

a) **(5%)** Lag en ny fil som inneholder funksjonen som starter spillet. Du kan enten lage en ny *main*-funksjon, eller la funksjonen hete *playMastermind* og kalle den fra *main*-function i deloppgave 1.

Hint: Definere følgende globale heltallskonstanter: SIZE (antall tegn i koden, 4), LETTERS (antall forskjellige bokstaver, 6). Dette gjør det enklere å forandre programmet senere, og lettere å lese koden din. Merk deg at alle konstanter bør skrives med store bokstaver (mens variabler består av flest små) Viktig: Du bør alltid prøve å unngå å bruke globale *variabler* siden programmet ditt da er særlig utsatt for feil og blir vanskelig å lese for andre.

b) **(10%)** Skriv funksjonen *randomChar* som returnerer en tilfeldig bokstav (char) i området definert av LETTERS konstanten. (F.eks. hvis LETTERS er satt til 3, skal funksjonen returnere 'A', 'B' eller 'C')

Hint: Tegn (char) er det samme som heltall i C/C++, og du kan typekaste direkte mellom dem. Koden cout « (char)65; skriver ut 'A'. 65+1 tilsvarer B osv. Du skal med andre ord finne et tilfeldig tall mellom 65 og 65+LETTERS-1 og returnere det som en char.

c) **(10%)** Lag en tabell kalt *code* med lengde SIZE og bruk funksjonen fra **b)** for å fylle tabellen med tilfeldige bokstaver. Dette er koden spilleren skal prøve å gjette.

d) **(10%)** Lag en tabell kalt *guess*. Denne tabellen skal inneholde bokstavene spilleren gjetter. Skriv funksjonen *readInput* som tar en tabell som argument. Funksjonen skal spørre spilleren etter SIZE bokstaver, og lagre dem i tabellen som ble gitt som argument.

Valgfritt: Skriv en funksjon som sjekker om bokstaven spilleren skrev inn er gyldig (ligger i området 65 - 65+LETTERS-1), så det er umulig å gjette bokstaver som ikke kan være i koden. Bruk denne funksjonen til å sjekke bokstavene brukeren gir i readInput-funksjonen hver gang denne blir kalt.

e) **(10%)** Skriv de to funksjonene *checkCharacters* og *checkCharactersAndPosition*. Den første skal returnere hvor mange riktige bokstaver spilleren har gjettet, og den andre hvor mange riktige bokstaver spilleren har plassert på rett plass. Begge skal returnere svaret som heltall. Utvid koden din fra **d)** slik at programmet ditt spør spilleren etter en ny kode så lenge *checkCharactersAndPosition* returnerer et tall mindre enn SIZE.

Hint: Det er flere måter du kan implementere checkCharacters. En måte er å telle antall 'A'er i både *code* og *guess* tabellene. Antall riktig gjettede 'A'er er da det laveste antallet 'A'er i *code* eller *guess*. Ved å gjøre det samme for 'B', 'C' og så videre, kan man få det totale antall riktig gjettede bokstaver.

f) **(5%)** Skriv funksjonen *printCode* som skriver ut en tabell med koden (den originale, eller spillerens gjetting).

g) **(10%)** Til slutt, sett sammen alle funksjonene i funksjonen fra **a)**. Når du er ferdig skal programmet lage og lagre en tilfeldig kode som spilleren kan gjette på inntil han finner den rette koden. For hver kode spilleren gjetter skal programmet skrive ut hvor mange rette bokstaver spilleren gjettet, og hvor mange av dem som var på rett plass.

Hint: Ved å bruke funksjonen printCode til å skrive ut tabellen med den tilfeldige koden i starten av programmet ditt vil det bli lettere å teste og feilsøke i koden din.

Valgfritt: Utvid koden din slik at spilleren har et begrenset antall forsøk på å gjette koden.