Introduktion till C-programmering

F1 och F2

Övningar till F1 och F2

1. Skriv ett program som läser tecken från standard input och räknar antalet meningar. Meningar avslutas med punkt, utropstecken eller frågetecken.
2. Skriv ett program som kopierar standard input till standard output. Vid kopieringen skall all text omgiven av < och > inklusive dessa tecken utelämnas.
3. Skriv ett program som läser tecken från standard input och räknar antalet ord. Med *ord* menas en obruten följd av bokstäver.
4. Modifiera programmet ovan så att det också skriver ut hur många bokstäver det är i det längsta ordet.
5. Skriv en C-funktion för att beräkna den harmoniska summan

   $$1 + 1/2 + 1/3 + 1/4 ... + 1/n$$

   Vilka parametrar och och vilken returtyp bör funktionen ha?
6. Skriv ett C-program som tabulerar ovanstående summa för $n = 1, 2, \ldots, 10$

Övningar till F1 och F2 (forts)

7. Skriv ett C-program räknar ut skriver hur många tal som behövs för att summan

$$1 + 1/2 + 1/3 + 1/4... + 1/n$$

   skall överstiga 10.

8. Skriv en funktion `int isEqual(char c1, char c2)` som returnerar 1 om tecknen `c1` och `c2` är lika annars 0. Om tecknen är bokstäver skall de betraktas som lika oavsett om de är versaler eller gemena (dvs 'a' och 'A' skall betraktas som lika).

9. Skriv ett program som läser en rad från standard input och skriver ut raden översatt till rövarspråket. Översättningen tillgår så att om `x` är en konsonant så ersätts den med `xox` medan vokaler lämnas oförändrade. Exempel: Texten "Don't panic" blir "Dodonon'tot popanonicoc".

Övningar till F1 och F2 (forts)

10. Skriv en funktion som läser en rad av godtycklig längd från standard input och skriver ut raden baklänges så att det sist inlästa tecknet skrivs först. Funktionen behöver inte använda någon array eller lista.
11. Följden

    $$0, 1, 1, 2, 3, 5, 8, 13 \ldots$$

    kallas Fibonaccital.

    a) Skriv en funktions `int fib(int n)` som beräknar och skriver ut det n:te Fibonaccitalet
    b) Skriv ett program som läser in ett tal n samt beräknar och skriver de n första Fibonaccitalen.
    c) Skriv ett program som läser in ett tal m samt beräknar hur många av Fibonaccitalen som är mindre än eller lika med m.
12. Skriv en rekursiv funktion `void printb(int x, int b)` som skriver ut `x` i basen `b`. Förutsätt för enkelhetens skull, att $b <= 10$.

Arrayer

F4

Övningar

1. Skriv en funktion **double** `enorm(`**double** `x[], `**int** `n)` som beräknar och returnerar den euklidiska vektornormen för vektorn x med n element. Vektornormen av vektorn $x_1, x_2, \ldots x_n$ definieras som $\sqrt{\sum_{i=1}^{n} {x_i}^2}$
2. Skriv funktionerna **int** `isVowel(`**char** `c)` och **int** `isConsonant(`**char** `c)` som returnerar 1 om argumetet `c` är en *vokal* respektive en *konsonant* annars 0. Skriv också ett huvuprogram som läser tecken från standard input och räknar antalet vokaler respektive konsonanter.
3. Implementera funktionen `readWord` enligt diskussionen ovan. Skriv också ett testprogram till funktionen.
4. Skriv ett program som läser standardinput och skriver ut det längsta ordet.
5. Skriv ett program som läser standard input och räknar (engelska) bokstävernas förekomstfrekvens. Ledning: Använd en heltalsarray med 26 platser. Använd plats 0 för att räkna *a*, plats 1 för *b* etc.

6. Standardfunktionen `strcpy` kontrollerar inte att utrymmet för mottagande område är tillräckligt stort. Skriv en egen funktion `stringCopy` som gör detta. Vilka parametrar behövs? Vad skall funktionen göra om det mottagande området är för litet?

7. En array innehåller heltal sorterade i storleksordning. Arrayen har en viss deklarerad storlek men det är inte säkert att hela utnyttjas. Man måste således också hålla reda på hur många tal som verkligen är lagrade. Skriv en funktion som lägger in ett nytt tal i arrayen så att sorteringsordningen bibehålls. Vilka parametrar är lämpliga? Hur skall man hålla reda på storleken respektive antal lagrade element?

   Skriv också en funktion för sökning efter ett givet värde.

8. Skriv en funktion `int match(char m[], char s[])` som returnerar index för första förekomst av strängen `m` i strängen `s`. Om strängen inte finns skall `-1` returneras.

9. Skriv en funktion `int nMatch(char m[], char s[])` som räknar och returnerar *antalet förekomster* av strängen `m` i strängen `s`.

Pekare

F5

Övningar

1. Skriv en funktion som löser andragradsekvationen

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

Lösningen ges av

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Koefficienterna a, b och c skall skickas till funktionen. Skriv tre olika varianter för att returnera lösningen:

a) via två parametrar,
b) som en post (funktionsvärde) och
c) som en pekare till en post.

Skriv också ett huvudprogram som testar funktionen med en följd av inlästa koefficienter.

2. Följande post är avsedd att lagra ett varierande antal heltal

```
struct container {
  int *buffer; /* Pekare till lagringsarea */
  int size; /* Allokerad storlek */
  int number; /* Aktuellt antal */
};

typedef struct container container;
```

   Skriv en funktion `int store(int data, container *c)` som lagrar `data` på första lediga plats och returnerar dess index. Om behållaren är full (dvs om `size==number`) så skall en ny, dubbelt så stor buffer anskaffas och innehållet i den gamla flyttas över till den nya.

   Skriv också en funktion som returnerar talet som är lagrat på en viss plats. Hur bör parametrarna till denna funktion se ut?

3. Samma problem som ovan *men* med lagring av textsträngar i stället för heltal.

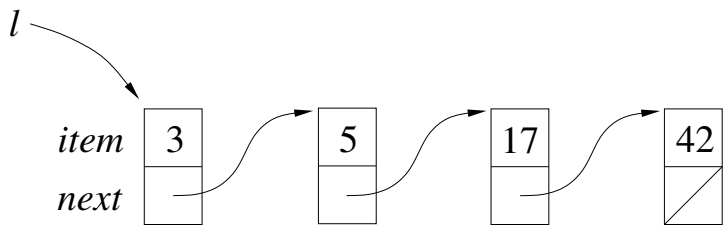
Länkade strukturer

F7

Länkade (rekursivt definierade) strukturer

Byggs med hjälp av poster (**`struct`**) som innehåller pekare till poster av samma typ.

Exempel: Länkad lista med heltal




```
typedef struct listElem {
    int item;
    struct listElem *next;
} listElem, *Link;
```

Övningar på länkade listor

1. Skriv en funktion `int length(Link link)` som returnerar längden av listan vars första nod pekas ut av `link`. Skriv både en iterativ och en rekursiv variant!
2. Skriv en funktion `int search(int item, Link link)` som returnerar 1 om `item` finns i listan som börjar vid `link`, annars 0. Skriv både en iterativ och en rekursiv variant!
3. Skriv en funktion `insert` som, givet en pekare till första noden i en lista där elementen är sorterade i storleksordning, lägger in ett nytt element i listan så att sorteringen bibehålls. Gör först en variant som gör en ny lista och sedan en variant som modifierar den befintliga listan.
4. Skriv en funktion `void addLast(int itm, link *lastPointer)` som lägger in ett nytt element med `item` sist i listan som pekas ut av `*lastPointer`.
5. Skriv en funktion som tar bort alla element med ett givet innehåll.
6. Skriv en funktion `Link merge(Link l1, Link l2)` som, givet att `l2` och `l2` pekar till första noden i var sin sorterade lista, skapar och returnerar en ny sorterad lista bestående av elementen från både `l2` och `l2`.

Övningar på binära träd

1. Deklarera en post för att representera noder i ett binärt träd och skriv en konstruktor (cons-funktion) för sådana poster.
2. Skriv funktioner som beräknar antalet noder respektive höjden i ett sådant träd. Funktionerna skall få en pekare till rotnoden som parameter.
3. Skriv en funktion avgör om två träd är isomorfa (har samma struktur).
4. Skriv en funktion som skriver ut ett träd i preorder med en nod per rad och en indentering som visar strukturen (dvs proportionell mot nodens djup)

Övningar

1. Skriv en funktion `int nextChar(FILE *f)` som läser förbi "white space" (blank, tab och radbyte) och returnerar det tecknet som följer. Tecknet skall läggas tillbaka i strömmen så att nästa läsning börjar vid det tecknet. Vid filslut skall `EOF` returneras.
2. Skriv ett program för att lista filer med angivande av radnummer. Programmet skall anropas med angivande av filnamn. Exempel:

   ```
   marvin$ list uppgift1.c
   ... output ...
   marvin$
   ```

   Om filen inte hittas så skall ett felmeddelande skrivas.