

## Programmeringsteknik med C och Matlab

---

**Skrivtid:** 16-20.

**Hjälpmedel:** Pennor.

**Antal uppgifter:** 7.

**Instruktioner. OBS! Läs igenom instruktionerna noggrant innan du börjar lösa uppgifterna.**

- Börja med att skriva ditt namn och personnummer på första bladet. Skriv sedan ditt kodnummer (från försättsbladet) **på varje blad**.
- Skriv i mån av plats dina lösningar direkt i tentamen.
- Om du behöver mer plats, skriv dina lösningar på extrablad. Lös bara en uppgift på varje extrablad. Se till att ditt **kodnummer** och **uppgiftens nummer** är tydligt markerade på varje extrablad du lämnar in.
- Se till att den C-kod som ingår i dina lösningar är vettigt indenterad.
- **Skriv tydligt.** Om vi inte kan läsa dina lösningar kan vi inte ge poäng för dem.
- Observera att tentamen är tryckt dubbelsidig. Läs alltså båda sidorna av varje blad.

**Betygsättning:** Totalt går det maximalt att uppnå 44 poäng.

- För betyget **3** (godkänt) krävs **22 poäng**.
- För betyget **4** krävs **29 poäng**.
- För betyget **5** krävs **36 poäng**.

Kodnummer: \_\_\_\_\_

## Uppgift 1

(5 Poäng)

Betrakta följande C-program:

```
#include <stdio.h>

int main(void){
    double x = 5;
    int y = -3;
    char c = 'd';
    double z = y + 4.55;
    int w = 4;

    printf("%f\n", x); /* Utskrift 1 */

    printf("%d\n", w/3); /* Utskrift 2 */

    x += z;

    printf("%f\n", x); /* Utskrift 3 */

    printf("%.2f\n", w - 1.0); /* Utskrift 4 */

    printf("%c %c %d\n", c, c, y); /* Utskrift 5 */
    return 0;
}
```

Vad skriver programmet ut när det körs?

Utskrift 1:	5.000000
Utskrift 2:	1
Utskrift 3:	6.550000
Utskrift 4:	3.00
Utskrift 5:	d d -3

Σ:

Kodnummer: \_\_\_\_\_

---

## Uppgift 2

(3 Poäng)

En cylinders volym är dess basyta gånger dess höjd. Basytan är i sin tur radien i kvadrat gånger  $\pi$ . Skriv en C-funktion som givet höjden och radien som inparametrar returnerar motsvarande cylinders volym. Använd närmrevärdet 3.1416 för  $\pi$ . Din funktion ska ha följande prototyp:

```
double cylinderVolume(double height, double radius);
```

Lösningsförslag:

```
double cylinderVolume(double height, double radius){  
    return 3.1416 * radius * radius * height;  
}
```

$\Sigma$ :

Kodnummer: \_\_\_\_\_

---

### Uppgift 3

(7 Poäng)

a) Skriv en funktion med prototypen

```
void absoluteValues(int source[], int absValues[], int n);
```

som lagrar absolutvärdena av de första  $n$  talen i arrayen `source` på de första  $n$  platserna i arrayen `absValues`.

(Absolutvärdet av ett tal  $x$  är dess avstånd från noll. Absolutvärdet av  $x$  är alltså  $x$  om  $x$  är positivt eller noll och  $-x$  om  $x$  är negativt.)

(3 poäng)

Lösningsförslag:

```
void absoluteValues(int source[], int absValues[], int n){
    int i;
    for(i = 0 ; i < n ; i++){
        if(source[i] >= 0) absValues[i] = source[i];
        else absValues[i] = -source[i];
    }
}
```

b) Skriv en main-funktion som

1. deklarerar två `int`-arrayer, `array1` och `array2`, vardera av längd 21,
2. fyller `array1` med talen  $-10, -9, \dots, -1, 0, 1, \dots, 9, 10$ ,
3. använder funktionen `absoluteValues` för att fylla `array2` med absolutvärdena av talen i `array1`, och
4. skriver ut talen i `array2`, ett på varje rad.

(4 poäng)

Lösningsförslag:

```
int main(void){
    int array1[21], array2[21];
    int i;

    for(i = 0 ; i < 21 ; i++)
        array1[i] = i -10;
    absoluteValues(array1, array2, 21);
    for(i = 0 ; i < 21 ; i++)
        printf("%d\n", array2[i]);
    return 0;
}
```

$\Sigma:$

Kodnummer: \_\_\_\_\_

---

#### Uppgift 4

(7 Poäng)

Betrakta följande C-funktioner:

---

```
int turnip(int carrot, int parsnip){
    if(carrot <= parsnip && parsnip > 2){
        return carrot + parsnip;
    }else{
        return parsnip + 1;
    }
}
```

```
int earth(int wind, int water){
    int fire = -water;
    int i = 0;
    do{
        i++;
        fire += wind;
    }while(i <= water);
    return fire;
}
```

---

Vilket returvärde ger följande tre anrop?

turnip(2,3);	5
turnip(2,2);	3
earth(4,2);	10

$\Sigma$ :

Kodnummer: \_\_\_\_\_

---

### Uppgift 5

(7 Poäng)

Betrakta följande C-kod:

---

```
void printDate(date d){
    printf("%d-%2d-%2d", d.year, d.month, d.day);
}

void printEmployeeData(employee e){
    printf("Name: %s %s\n", e.firstName, e.lastName);
    printf("Date of birth: ");
    printDate(e.birthDate);
    printf("\n");
    printf("Employed since: ");
    printDate(e.employmentDate);
    printf("\n");
    printf("Salary: %.2f kr\n",e.salary);
}

int main(void){
    date employmentDate,birthDate;
    employee e;
    employmentDate.year = 1998;
    employmentDate.month = 5;
    employmentDate.day = 24;
    birthDate.year = 1975;
    birthDate.month = 10;
    birthDate.day = 2;
    strcpy(e.firstName, "Sara");
    strcpy(e.lastName, "Smith");
    e.birthDate = birthDate;
    e.salary = 25342.23;
    e.employmentDate = employmentDate;

    printEmployeeData(e);
    return 0;
}
```

---

a) Definiera datatypen **date** som används i ovanstående kod (3 poäng).

Lösningsförslag:

```
typedef struct{
    int year;
    int month;
    int day;
} date;
```

b) Definiera datatypen **employee** som används i ovanstående kod (4 poäng).

Lösningsförslag:

```
typedef struct{
    char firstName[100];
    char lastName[100];
    date birthDate;
    date employmentDate;
    double salary;
} employee;
```

Σ:

Kodnummer: \_\_\_\_\_

---

## Uppgift 6

(10 Poäng)

a) Skriv en funktion

```
double * allocDoubleArray(int length);
```

som allokerar en double-array av längd `length` och returnerar en pekare till arrayen. (3 poäng)

Lösningsförslag:

```
double * allocDoubleArray(int length){
    return (double *)calloc(length, sizeof(double));
}
```

b) Skriv en funktion

```
void sumPosAndNeg(double * array, int length, double * posSum, double * negSum);
```

som

- lagrar summan av alla *positiva* tal bland de `length` första elementen i `array` i variabeln som pekaren `posSum` pekar på, och
- lagrar summan av alla *negativa* tal bland de `length` första elementen i `array` i variabeln som pekaren `negSum` pekar på.

(4 poäng)

Lösningsförslag:

```
void sumPosAndNeg(double * array, int length, double * posSum, double * negSum){
    int i;

    *posSum = 0;
    *negSum = 0;
    for(i = 0 ; i < length ; i++){
        if(array[i] > 0) *posSum += array[i];
        else *negSum += array[i];
    }
}
```

c) Givet ett tal  $n$  är pyramidtalet för  $n$ , skrivet  $pyramid(n)$ , summan av alla positiva heltal som är mindre än eller lika med  $n$ . Alltså:

$$pyramid(n) = 1 + 2 + \dots + n = \sum_{i=1}^n i$$

Skriv en *rekursiv* funktion

```
int pyramid(int n);
```

som givet talet  $n$  som inparameter returnerar det motsvarande pyramidtalet.

(3 poäng)

Lösningsförslag:

```
int pyramid(int n){  
    if(n <= 1) return 1;  
  
    return n + pyramid(n-1);  
}
```

$\Sigma$ :

Kodnummer: \_\_\_\_\_

---

### Uppgift 7

(5 Poäng)

Bestäm vilka av följande utsagor som är sanna och vilka som är falska. För varje riktigt svar ges 1 poäng, för varje felaktigt svar -1 poäng. Om totalsumman för uppgiften blir negativ ges totalt 0 poäng.

- |  | sant                     | falskt                   |
|--|--------------------------|--------------------------|
| a) Uttrycket $i \neq j$ ; gör samma sak som uttrycket $i = i/j$ ; <b>Sant.</b>   | <input type="checkbox"/> | <input type="checkbox"/> |
| b) C tolkar alla kommandoradsparametrar till ett program som strängar. <b>Sant.</b>  | <input type="checkbox"/> | <input type="checkbox"/> |
| c) Om <code>a</code> är en <code>int</code> -variabel så är <code>&amp;a</code> adressen till <code>a</code> . <b>Sant.</b>  | <input type="checkbox"/> | <input type="checkbox"/> |
| d) Funktionen <code>calloc</code> returnerar antalet element i arrayen den allokerat. <b>Falskt.</b> ( <code>calloc</code> returnerar en pekare till det minne den allokerat.) | <input type="checkbox"/> | <input type="checkbox"/> |
| e) En <i>funktionsdefinition</i> måste alltid föregås av en <i>funktionsdeklara-tion</i> . <b>Falskt.</b>  | <input type="checkbox"/> | <input type="checkbox"/> |

Σ: