Tentamen - Programmering

DVA117

----------FACIT----------

Akademin för innovation, design och teknik

Torsdag 2019-08-15

An English translation of the entire exam follows after the questions in Swedish

Skrivtid: 08.10 – 13.30
Hjälpmedel: Valfritt icke-elektroniskt material
Lärare: Caroline Uppsäll
(kan nås på telefon om du frågar tentavakten)

Preliminära betygsgränser

Betyg 3:	18p
Betyg 4:	25p
Betyg 5:	30p
Max:	**34p**

Allmänt

- All kod skall skrivas i C.
- Skriv tydligt vilken uppgift/deluppgift ditt svar anser.
- Skriv endast på bladets **ena** sida, börja ny uppgift på nytt papper.
- Referera inte mellan olika uppgifter.
- Om du är osäker på vad som avses i någon fråga, skriv då vad du gör för antagande.
- Oläsliga/oförståeliga/ostrukturerade svar rättas inte.
- Kommentera din kod.
- Det är inte tillåtet att använda goto-satser och globala variabler
- Tips: Läs igenom hela tentamen innan du börjar skriva för att veta hur du ska disponera din tid.
- Potentiellt intjänade bonuspoäng kan endast användas på kursinstansens ordinarie tentamen, alltså inte på denna tentamen.
- Förklaring som till viss del är rätt och till viss del är fel ger inga poäng. Otillräcklig förklaring/beskrivning ger inga poäng.

Lycka till!
/Caroline & Robert

----------FACIT----------

Notera: alla lösningar bedöms individuellt och förändringar från nedanstående poängsättning (inom uppgiften) kan förekomma beroende på hur resterande svar ser ut.

Uppgift 1 [1p]

Beskriv en fördel med dynamiskt allokerat minne.

Svar: Minne allokerat i en funktion behålls även när man går ur funktionen (block). Dynamiskt, dvs. man kan allokera exakt så mycket minne som behövs vid olika tillfällen under programmets exekvering.

Uppgift 2 [1p]

Vad evaluerar följande uttryck till?

```
int x=4, y=26, z=45, a=1, b=9;


!(a == x || a != y) && (b < x && a)
```

Svar: 0 / Falskt

Uppgift 3 [3p]

Vilka av följande påståenden är sanna?

a) &-operatorn är innehållsoperatorn som ger oss innehållet på en adress.
b) Lokala variabler lagras på stacken.
c) Heltal och tecken kan sparas i samma array (som int respektive char).
d) Dynamiskt allokerat minne lagras på stacken.
e) &-operatorn är adressoperatorn och ger oss adressen där en variabel finns i minnet.
f) Om en funktion inte ska returnera något data via return-satsen så används void som returtyp.
g) För att jämföra om två strängar är lika kan man använda ==.
h) En pekarvariabel innehåller en adress.
i) Funktionen strlen ger längden på en sträng utan strängslutstecknet (’\0’).
j) Syntax är den uppsättning grammatiska regler som gäller för programspråket.
k) En while-loop kör loopkroppen alltid minst en gång.
l) En strukt kan maximalt innehåller 5 fält.
m) En switch-sats evaluerar på resultatet av ett villkor (sant eller falskt) medan if-satsen evaluerar på ett heltalsvärde.

Svar b e f h i j

Uppgift 4 [4p]

I nedanstående program har ett antal fel smygit sig in. Din uppgift är att hitta och rätta felen. Skriv av det radnummer där felet finns följt av den korrekta satsen.

```
1       #include <stdio.h>
2
3       void countCharacters(char *str, int *nr);
4
5       int main(void)
6       {
7
8           char input[100];
9           int nrOfCharacters = 0;
10          printf("Please enter a sentence: ");
11          fgets(&input, 100, stdin);
12
13          countCharacters(input, &nrOfCharacters);
14
15          printf(“%c characters in the string\n", nrOfCharacters);
16          return 0;
17      }
18
19      void countCharacters(char *str, int *nr)
20      {
21
22          for(i = 0; i != '\0'; i++);
23
24          nr = i;
25      }
```

Svar:
11 – input är redan en pekar och vi ska därför inte ange &-tecknet vid inläsning.
15 – nrOfCharacters är ett heltal och ska skrivas ut med %d (inte %c)
21 eller 22 – i är inte deklarerad.
22 – det är str[i] som ska jämföras med ’\0’, inte i (i är bara en räknare)
*24 – nr är en pekare och det är det den pekar på som ska uppdateras till i, alltså *nr*

Uppgift 5 [7p]

a) *(2p)* Skapa en egen typ (strukt/post) som heter ”`employee`” och som innehåller information om en anställd. Den information som ska finnas i typen är följande:

- Förnamn – max 25 tecken
- Efternamn – max 25 tecken
- Födelsedatum – sparat på formatet ÅÅMMDD
- Timlön
- Arbetade timmar innevarande månad (ska anges som timmar hela timmar)

b) *(2p)* Låt användaren ange hur många anställda som ska finnas och skapa sedan en dynamiskt allokerat arrat (`employeeList`) som räcker för det angivna antalet anställda. Glöm inte att testa så att allokeringen lyckas.

c) *(3p)* Skriv en funktion som låter användaren beräkna hur mycket företaget ska betala i lön totalt den aktuella månaden. En anställds lön beräknas genom att multiplisera timlönen med antalet arbetade timmar. Den totala lönen för samtliga anställda (employeeList) ska returneras ur funktionen. Funktionens returyp måste vara void, vilka parametrar som behövs bestämmer du själv.

Ange också hur anropet till funktionen ser ut .

Du kan anta att samtliga anställdas information är korrekt angiven.

Svar:

a)

```
struct employee
{
     char firstName[25];
     char lastName[25];
     int dateOfBirth;
     float hourlySalary;
     int hoursWorked;
};
```

b)

```
int nrOfEmployees;
printf("Enter nr of employees: ");
scanf("%d", &nrOfEmployees);

struct employee *employeeList =
           (struct employee*)malloc(nrOfEmployees *
                                    sizeof(struct employee));
if(employeeList == NULL)
    printf("Error allocating");
```

c)

```
void totalSalary(struct employee *list, int nr, float *total)
 {
     *total = 0;

     for(int i = 0; i < nr; i++)
         *total = *total + list[i].hoursWorked *
                           list[i].hourlySalary;
 }

 Anrop:
          float tot;
          totalSalary(employeeList, nrOfEmployees, &tot);
```

Uppgift 6 [5p]

Nedanstående program slumpar tal mellan 1 och 50. Om det slumpade talet är jämt skrivs det ut på skärmen följt av ett enterslag. När 5st jämna tal slumpats (och skrivits ut) så avslutas programmet. Du ska nu skriva om programmet med följande förändraing:

- Programmet ska slumpa tal i intervallet 10 till 60. *(1p)*
- Programmet ska skriva ut alla udda slumpade tal (istället för positiva). *(1p)*
- Programmet ska använda en do-while-loop istället för en while-lopp. *(1p)*
- Programmet ska avslutas när 10 udda tal slumpats fram (och skrivits ut). *(1p)*
- Programmet ska skriva ut ”Exiting program” samt antalet slumpade jämna tal (de som alltså slumpats fram men inte skrivits ut) innan programmet avslutas. *(1p)*

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {

    int randomNumber, nrOfEven = 0;
    srand(time(0));

    while(nrOfEven < 5)
    {
        randomNumber = rand()%50+1;
        if(randomNumber%2 == 0)
        {
            printf("%d\n", randomNumber);
            nrOfEven ++;
        }
    }
    return 0;
}
```

Svar:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {

    int randomNumber, nrOfOdd = 0, nrOfEven = 0;
    srand(time(0));

    do
    {
        randomNumber = rand()%50+10;
        if(randomNumber%2 == 1)
        {
            printf("%d\n", randomNumber);
            nrOfOdd++;
        }
        else
            nrOfEven++;
```

```
    }while(nrOfOdd < 10);
    printf("Exiting program, %d positiv numbers were generated\n",
nrOfEven);
    return 0;
}
```

Uppgift 7 [4p]

Skriv den kod som saknas i programmet nedan. Programmet ska ta reda på om ett tal (som användaren anger) är ett primtal eller inte samt skriver ut detta på skärmen.

Programmet accepterar endast positiva heltal större än 1 från användaren.

Ett primtal är ett tal som är jämnt delbart endast med sig själv och 1.

```
#include <stdio.h>

int main(void) {

    int number, notPrime = 0;
    do{
        printf("Enter a number to test: ");
        scanf("%d", &number);
    }while(number < 2);

    //Här saknas kod

    if(notPrime == 0)
        printf("%d is a prime\n", number);
    else
        printf("%d is not a prime\n", number);

    return 0;
}
```

Svar:

```
for(int i = 2; i < number; i++)
{
    if(number%i == 0)
    {
        notPrime = 1;
        break;
    }
}
```

Uppgift 8 [3p]

Vad har skrivits ut när nedanstående program kört färdigt?

```
#include <stdio.h>

int main(void)
{
    int a = 1, b = 2, i = 3;
    int *p = &a;
    while (i < 10) {
        a = *p;
        if (i > 6 || i == 3)
        {
            i += 2;
            p = &b;
        }
        else
        {
            i++;
            p = &a;
        }
        *p = *p + 1;
    }
    printf("a = %d\nb = %d\ni = %d\n", a, b, i);
    return 0;
}
```

Svar:

a=4
b=5
c=11

Uppgift 9 [4p]

Antag följande två struktdefinitioner

```
struct airplain
{
    char name[18];
    int seats;
    float fuelMaxLimit;
    int crewNeaded;
};

struct fleet
{
    struct airplain totalFleet[20];
};
```

Hur mycket minne tas i anspråk när…

a) typen airplain ovan definieras?

b) följande deklaration görs? `struct airplain myPlane;`
c) följande deklaration görs? `struct airplain *pMyPlane;`
d) följande deklaration görs? `struct fleet airline;`

Du kan anta att definitoner och deklarationer gör på ett standard 32-bitars system, alltså där en int tar upp 4 byte.

Svar:
a) 0 byte
b) 30 byte (18+4+4+4)
c) 4 eller 8 byte (en pekarvariabel)
d) 600 byte (30 * 20st)

Uppgift 10 [2p]

Vilket blir resultatet av följande program? Du kan anta att fopen lyckas.

```
#include <stdio.h>
#include <string.h>
#define SIZE 100

int main(void)
{
    char str[SIZE];
    FILE *fp;

    printf("Enter a string: ");
    fgets(str, SIZE, stdin);

    fp = fopen("theFile", "w");
    if(fp != NULL)
    {
        for(int i = strlen(str)-1; i >= 0; i--)
            fprintf(fp, "%c", str[i]);
        fclose(fp);
    }
    else
        printf("Error\n");

    return 0;
}
```

Svar: En fil som heter theFile som innehåller den inmatade strängen baklänges.

FACIT /SOLUTIONS

Exam - Programming

DVA117

----------Solutions----------

School of Innovation, design and technology

Thursday 2019-08-15

Writing time: 08.10 – 13.30
Aids: Any non-electronic material
Examiner: Caroline Uppsäll
(Can be reached by telephone if you ask the exam guard)

Preliminary grading limits

Grade 3: 18p
Grade 4: 25p
Grade 5: 30p
Max: **34p**

Generally

- All code should be written in C.
- Write clearly what task/sup-task your answers consider.
- Do only use **one** side of the paper, start new question on new paper.
- Do not refer between questions.
- If you are unsure of a meaning of a question, write down your assumption.
- Unreadable/incomprehensible answers will not be marked.
- Comment your code.
- It is not allowed to use goto-statements or global variables
- Hint: To know how to allocate your time, read through the entire exam before you start writing.
- Explanations that are to some extent correct and to som extent wrong does not give any points. Insufficient explanations/descriptions gives no points.

----------Solutions----------

Good luck!
/Caroline & Robert

---------- Solutions ----------

Note: all solutions are assessed invidually and changes from the following scoring (within the task) my occur depending on what the remaining answer looks like.

Question 1 [1p]

Describe an advantage of dynamically allocated memory.

Answer: Memory allocated in a function is retained when you exit the function (block). With dynamic memory you can also allocate exactly the amount of memory you need at any given time in the program.

Question 2 [1p]

What does the following expression evaluate to (true or false)?

```
int x=4, y=26, z=45, a=1, b=9;


!(a == x || a != y) && (b < x && a)
```

Answer: 0 / False

Question 3 [3p]

Which of the following statements are true?

a) The &-operator is the content operator that gives us the content at an address.
b) Local variables are stored on the stack.
c) Integers and characters can be stored in the same array (as int and char respectively).
d) Dynamically allocated memory is stored on the stack.
e) The &-operator is the address operator and gives us the address where a variable is stored in memory.
f) If a function does not return any data via the return statement, void is used as the return type.
g) To compare if two different strings are equal you can use ==.
h) A pointer variable contains an address.
i) The strlen function gives the length of a string without '\0'.
j) Syntax is the set of grammatical rules that apply to the program language.
k) The body of a while loop is always run at least once.
l) A structure can contain a maximum of 5 fields.
m) A switch statement is evaluated on if a condition is true or false.

Answer: b e f h i j

Question 4 [4p]

In the following program a number of errors occur. Your task is to find and correct the errors. Write the row number where you find an error followed by the correct line of code for that row.

```
1      #include <stdio.h>
2
3      void countCharacters(char *str, int *nr);
4
5      int main(void)
6      {
7
8          char input[100];
9          int nrOfCharacters = 0;
10         printf("Please enter a sentence: ");
11         fgets(&input, 100, stdin);
12
13         countCharacters(input, &nrOfCharacters);
14
15         printf("%c characters in the string\n", nrOfCharacters);
16         return 0;
17     }
18
19     void countCharacters(char *str, int *nr)
20     {
21
22         for(i = 0; i != '\0'; i++);
23
24         nr = i;
25     }
```

Answer:
11 – input is already a pointer, & is not needed. `fgets(input, 100, stdin);`
15 – nrOfCharacters is an integer and should be printed with %d, not %c
21 eller 22 – i is not declared.
22 – it is str[i] that should be compared to '\0', not i (i is only a counter)
24 – nr is a pointer and it is the data it's pointing to that should be updated. `*nr = i;`

Question 5 [7p]

a) *(2p)* Create your own type (struct) with the name "`employee`" that contains information about an employee. The type should contain the following:

- First name – max 25 characters
- Last name – max 25 characters
- Date of birth – Saved at the format YYMMDD
- Hourly salary
- Number of worked hours during the month (full hours).

b) *(2p)* Let the user specify how many employees there should be and then create a dynamically allocated array (`employeeList`) big enough for the specified number of employees. Do not forget to test so that the allocation is successful.

c) *(3p)* Write a function that lets the user calculate how much the company should pay in salary (total) the current month. An employee's salary is calculated by multiplying the hourly wage by the number of hours worked. The total salary for all employees (employeeList) must be returned from the function. The return type of the function must be void, what parameters you need is up to you to determine.

Also state what the function call looks like.

You can assume that all employees' information is correctly specified in the array.

Answer:

a)

```
struct employee
{
     char firstName[25];
     char lastName[25];
     int dateOfBirth;
     float hourlySalary;
     int hoursWorked;
};
```

b)

```
int nrOfEmployees;
printf("Enter nr of employees: ");
scanf("%d", &nrOfEmployees);

struct employee *employeeList =
           (struct employee*)malloc(nrOfEmployees *
                                    sizeof(struct employee));
if(employeeList == NULL)
    printf("Error allocating");
```

c)

```
void totalSalary(struct employee *list, int nr, float *total)
 {
     *total = 0;

     for(int i = 0; i < nr; i++)
         *total = *total + list[i].hoursWorked *
                           list[i].hourlySalary;
 }

 Anrop:
          float tot;
          totalSalary(employeeList, nrOfEmployees, &tot);
```

Question 6 [5p]

The program below genereates random numbers between 1 and 50. If the random number is even, it is printed on the screen followed by an enter stroke. When 5 even numbers are randomized (and printed), the program ends. You should now rewrite the program with the following change:

- The program should now generate random numbers between 10 and 60. *(1p)*
- The program should print all odd random numbers (instead of the positive ones). *(1p)*
- The program should use a do-while loop instead of a while loop. *(1p)*
- The program should end when 10 odd numbers are randomized (and printed) *(1p)*
- The program should print ”Exiting program” and the number om randomly generated even numbers (those that are randomized but not printed on the screen) before the program ends. *(1p)*

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {

    int randomNumber, nrOfEven = 0;
    srand(time(0));

    while(nrOfEven < 5)
    {
        randomNumber = rand()%50+1;
        if(randomNumber%2 == 0)
        {
            printf("%d\n", randomNumber);
            nrOfEven ++;
        }
    }
    return 0;
}
```

Answer:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {

    int randomNumber, nrOfOdd = 0, nrOfEven = 0;
    srand(time(0));

    do
    {
        randomNumber = rand()%50+10;
        if(randomNumber%2 == 1)
        {
            printf("%d\n", randomNumber);
            nrOfOdd++;
```

```
        }
        else
            nrOfEven++;
    }while(nrOfOdd < 10);
    printf("Exiting program, %d positiv numbers were generated\n",
nrOfEven);
    return 0;
}
```

Question 7 [4p]

Write the missing code in the program below. The program should find out if a number (which is specified by the user) is a prime number or not. This should also be printed on the screen.

The program only accepts positive integers higher than 1.

A prime number is a number that is evenly divisible only by itself and 1.

```
#include <stdio.h>

int main(void) {

    int number, notPrime = 0;
    do{
        printf("Enter a number to test: ");
        scanf("%d", &number);
    }while(number < 2);

    //Missing code goes here.

    if(notPrime == 0)
        printf("%d is a prime\n", number);
    else
        printf("%d is not a prime\n", number);

    return 0;
}
```

Answer:

```
for(int i = 2; i < number; i++)
{
    if(number%i == 0)
    {
        notPrime = 1;
        break;
    }
}
```

Question 8 [3p]

What will the following program print on the screen.

```
#include <stdio.h>

int main(void)
{
    int a = 1, b = 2, i = 3;
    int *p = &a;
    while (i < 10) {
        a = *p;
        if (i > 6 || i == 3)
        {
            i += 2;
            p = &b;
        }
        else
        {
            i++;
            p = &a;
        }
        *p = *p + 1;
    }
    printf("a = %d\nb = %d\ni = %d\n", a, b, i);
    return 0;
}
```

Answer:

a=4
b=5
c=11

Question 9 [4p]

Assume the following types:

```
struct airplain
{
    char name[18];
    int seats;
    float fuelMaxLimit;
    int crewNeaded;
};

struct fleet
{
    struct airplain totalFleet[20];
};
```

How much memory is used/allocated when…

a) the type airplain above is defined.
b) the following declaration is made? `struct airplain myPlane;`

c) the following declaration is made? `struct airplain *pMyPlane;`

d) the following declaration is made? `struct fleet airline;`

You can assume that definitions and declarations are made on a standard 32-bit system, ie where an int takes up 4 bytes.

Answer:

a) 0 byte

b) 30 byte (18+4+4+4)

c) 4 eller 8 byte (en pekarvariabel)

d) 600 byte (30 * 20st)

Question 10 [2p]

Which is the result of the following program? You can assume that fopen is successful.

```
#include <stdio.h>
#include <string.h>
#define SIZE 100

int main(void)
{
    char str[SIZE];
    FILE *fp;

    printf("Enter a string: ");
    fgets(str, SIZE, stdin);

    fp = fopen("theFile", "w");
    if(fp != NULL)
    {
        for(int i = strlen(str)-1; i >= 0; i--)
            fprintf(fp, "%c", str[i]);
        fclose(fp);
    }
    else
        printf("Error\n");

    return 0;
}
```

Answer: A file called theFile that contains the input string backwards.