Tentamen - Programmering

DVA117

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

Akademin för innovation, design och teknik

Måndag 2019-06-03

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.
- 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]

Förklara skillanden på följande två uttryck.

```
int number;
```

a) `number = 4`
b) `number == 4`

Svar: i a sker en tilldelning (tal sätts till 4) medan det i b sker en jämförelse (är tal 4?)
Om man i svaret säger att == innebär att number är 4 (utan vidare förklaring) är inte tillräckligt då det inte går att tolka vad det betyder.

Uppgift 2 [2p]

Vad innebär det att

a) en variabel deklareras?
b) en variabel initieras?
c) en funktion deklareras?
d) en funktion definieras?

Svara på vardera fråga med max 4 meningar.

Svar:

a) Minne tas i anspråk för variabeln (storlekn avgörs av typen) – variabeln skapas
   Ett svar som endast säger att variabeln skapas eller att den får ett namn eller en typ är inte tillräckligt. Kombinationer av dessa godkänns.
b) Ett startvärde sätts i variabeln
c) Funktionshuvudet är funktionsdeklarationen. Funktionsdeklarationen berättar vad funktionen heter, vilken typ den returnerar samt hur parameterlistan ser ut (typer och ordning).
   Ett svar som endast säger att funktionsdeklarationen berättar att funktionen finns är inte tillräckligt.
d) Funktionsdefinitionen är själva koden i funktionen – den kod som gör jobbet.

Uppgift 3 [1p]

Vad används &-operatorn respektive *-operatorn (förutom vid deklaration) till? Använd gärna ett exempel i din förklaring.

Svar: & används för att få adressen där en variabel ligger i minnet. *används för att få innehållet på en specifik adress i minnet.

Uppgift 4 [1p]

Vad saknas i nedanstående beräkning för att resultatet ska bli korrekt?

```
int totalSumOfPoints, nrOfStudents;
float averagePointsAtExam;
...
averagePointsAtExam = totalSumOfPoints/nrOfStudents;
```

Svar: en typkonvertering. Nu görs en division mellan två heltal vilket genererar ett heltal, vi förlorar decimaldelen av svaret.
Svar att inläsning och inkluderingar etc saknas bedöms inte som korrekt då det är själva beräkningen som frågan gäller.

Uppgift 5 [4p]

a) I nedanstående program – vad skapas på heapen och vad skapas på stacken? *(3p)*

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

int main(void) {

    int number = 5;
    float arrA[4] = {5, 10, 15, 20};
    int *pnumber = &number;
    float *arrB;
    arrB = (char*)malloc(5*sizeof(char));
    arrB[0] = 'a'; arrB[1] = 'b'; arrB[2] = 'c';
    return 0;
}
```

Skriv av följande tabell på ditt svarspapper och fyll i om variabeln/datat skapas/sparas på stacken eller på heapen.

	Stacken eller Heapen?
Variabeln: `number`	Stacken
Variabeln: `arrA`	Stacken
Datat: 5, 10, 15 & 20	Stacken
Variabeln: `pnumber`	Stacken
Variabeln: `arrB`	Stacken
Datat: 'a', 'b' & 'c'	Heapen

b) Hur länge existerar minne som allokeras/skapas på stacken respektive heapen? *(1p)*

Svar: Minne allokerat på stacken existerar i det block/skope {} där det variabeln deklarerats. Minne allokerat på heapen existerar tills programmet lämnar tillbaka det med anrop till funktionen free( );

Uppgift 6 [4p]

Nedan hittar du ett kort program där en del av programmet saknas (markerat med en ruta). Vardera av deluppgifterna nedan (a-c) anger en eller flera rader kod som ska läggas där kod saknas i programmet.

Nedan hittar du också ett antal förslag på utskrifter (1-6).

Din uppgift är att para ihop varje kodblock (a-c) med rätt utskrift (1-6), det kan finnas utskrifter som är korrekta för flera av kodsekvenserna och det finns utskrifter som inte stämmer med någon av kodsekvenserna.

```
#include <stdio.h>

int main(void) {

    int x = 0, y = 0;
    while(x < 5)
    {
        [          ]

        printf("%d%d  ", x, y);
        x = x + 1;
    }

    return 0;
}
```

Förslag på utskrifter

1) 11 23 36 410
2) 13 34 53
3) 13 34 53 70
4) 11 34 59
5) 02 14 25 36 47
6) 01 12 24 36 48

a) (1p)

```
x = x + 1;
y = y + x;
```

b) (1p)

```
y = y + 2;
if(y > 4)
      y = y - 1;
```

c) (2p)

```
if(y < 5)
{
    x = x + 1;
    for(int i = 0; i < x; i++)
          y = y - 1;
}
y = y + 4;
```

Svar: a – 4 b – 5 c – 2

Har man svarat att två alternativ stämmer på en kodsekvens och det ena alternativet är rätt så ges inte full poäng – detta ses som en gardering av svaret. Ett program med fasta värden på variabler kan inte ge olika utskrifter.

Uppgift 7 [4p]

Du ska nu skriva en funktion, `countCharacters`, vars uppgift är att ta in en sträng och räkna antalet tecken i strängen exklusive strängslutstecknet (’\0’) – du ska med andra ord göra din egna strlen-funktion.

Du ska utgå från nedanstående program.

```
int main(void) {

    char input[100];
    int nrOfCharacters = 0;
    printf("Please enter a sentence: ");
    fgets(input, 100, stdin);

    /*Här ska funktionen countCharacters anropas*/

    printf("The sentence consists of %d characters\n", nrOfCharacters);
    return 0;
}
```

Funktionen ska heta `countCharacters` och ha returtyp void, vilka parametrar funktionen behöver ha är det upp till dig att avgöra. Funktionens uppgift är att räkna antalet tecken i en angiven sträng (i exemplet ovan är det alltså tecknena i strängen `input` som ska räknas). Strängslutstecknet ’\0’ ska inte räknas med. Notera att antalet tecken i strängen ska skrivas ut i main. Du får inte använda `strlen( )` i din lösning. *(3p)*

fgets() läser även in det avslutande enterslaget till strängen, du väljer själv om detta ska räknas med i antalet tecken eller inte.

Skriv också själva anropet till funktionen, anropet ska ske där kommentaren i koden ovan ligger. *(1p)*

Lösningsförslag:

Anrop: `countCharacters(input, &nrOfCharacters);`

```
void countCharacters(char *str, int *nr)
{
    int i;
    for(i = 0; str[i] != '\0'; i++);

    *nr = i;

}
```

Uppgift 8 [2p]

Titta på nedanstående program

```
#include <stdio.h>
#define SIZE 5

int* update(int array[])
{
    for(int i = 0; i < SIZE; i++)
        array[i] = array[i] + 1;
    return array;
}

int main(void) {
    int arr[SIZE] = {1, 2, 3, 4, 5};
    arr = update(arr);

    for(int i = 0; i < SIZE; i++)
        printf("%d  ", *(arr+i));
    return 0;
}
```

Avgör om programmet går igenom kompilatorn eller inte.

Om du anser att programmet går igenom kompilatorn – vilken blir utskriften?
Om du anser att programmet inte går igenom kompilatorn – förklara varför samt hur man kan lösa problemet.

Svar: Programmet kompilerar inte.
Arrayen arr i main kan inte tilldelas. Vill man ändra i en array i en annan funktion än där arrayen deklarerats så räcker det att skicka in arrayen till funktionen.
Funktionshuvudet bör alltså vara: void update(int array[ ]); och anropet: update(arr);

Uppgift 9 [12p]

Du ska i nedanstående deluppgifter bygga en del av ett program som hanterar ett akvarium.

a) Definiera en egen typ som heter `fish` och som kan hålla följande information om en fisk: *(1p)*

- ras (`species`) – t.ex. ”Piraya” eller ”Clownfisk”
- ålder (age)
- antal tänder (`teeth`)

Svar:

```
struct fish
{
    char species[20];
    int age;
    int teeth;
};
```

b) Deklarera en variabel ”Snappy” av den egna typ du skapade i deluppgfit a). Sätt rasen (`species`) till ”Piranha”, ålder (age) till 4 och antal tänder (`teeth`) till 69. *(2p)*

Här kan funktionen `strcpy(char *dst, char *src);` vara till hjälp.

Svar:

```
struct fish snappy;
strcpy(snappy.species, "Piranha");
snappy.age = 4;
snappy.teeth = 69;
```

Initiering direkt vid variabeldeklaration är också ok.

c) Deklarera nu en pekare ”`aquarium`” av den egna typen från deluppgift a). Låt pekaren peka på ett dynamiskt allokerat minne som är tillräckligt stort för att hålla information om 10 fiskar. Visa också hur du testar om allokeringen lyckades eller inte. *(2p)*

Svar:

```
struct fish *aquarium = NULL;
aquarium = (struct fish*)calloc(10, sizeof(struct fish));
if(aquarium == NULL)
     printf("Error allocating");
else
{
    //jobba med minnet
}
```

d) Skriv en funktion som heter `fillAquarium` som tar in en pekare till akvariet (det dynamiskt allokerade minnet i deluppgift c) samt antalet platser i akvariet och som låter användaren ange information om alla fiskarna (`species`, `age` och `teeth`). Returtypen ska vara void.

Skriv också hur anropet till funktionen skulle se ut om det är det dynamiska `aquarium` från c) som ska fyllas med information. *(3p)*

Svar:

```
void fillAquarium(struct fish *aq, int size)
{
     for(int i = 0; i < size; i++)
     {
          printf("Enter species: ");
          gets(aq[i].species);
          printf("Enter age: ");
          scanf("%d", &aq[i].age);
          printf("Enter number of teeth: ");
          scanf("%d", &aq[i].teeth);
      }
}
```

Anrop: `fillAquarium(aquarium, 10);`

e) Skriv en funktion som heter `addFishes` vars uppgift är att öka hur många fiskar som får plats i akvariet – alltså öka mängden dynamiskt allokerat minne. Det ny antalet fiskar ska skickas in till funktionen, övriga parametrar samt returtyp bestämmer du själv. Om det inte går att öka storleken på akvariet (mängden minne) ska det gamla akvariet returneras/finnas kvar.

Visa också hur anropet till funktionen ser ut då arrayen `aquarium` ska utökas till totalt 20 fiskar. *(4p)*

Svar:

```
struct fish* addFishes(struct fish *aq, int newSize)
{
    struct fish* temp = (struct fish*)realloc(aq, newSize *
                                        sizeof(struct fish));
    if(temp != NULL)
         return temp;
    else
    {
         printf("Error");
         return aq;
    }
}
```

Anrop: `aquarium = addFishes(aquarium, 20);`

ELLER

```
void addFishes(struct fish **aq, int newSize)
{
     struct fish* temp = (struct fish*)realloc(*aq, newSize *
                                                sizeof(struct fish));
     if(temp != NULL)
         *aq = temp;
     else
         printf("Error");
}
```

Anrop: `addFishes(&aquarium, 20);`

Uppgift 10 [3p]

I nedanstående program saknas 4 rader kod som hanterar kommunikation med en extern fil. Din uppgift är att skriva de fyra raderna.

```
#include <stdio.h>
#define SIZE 10

int main(void) {

    int numbers[SIZE];
    for(int i = 0; i < SIZE; i++)
    {
        printf("Enter number %d: ", i+1);
        scanf("%d", &numbers[i]);
    }

    // a) Deklarera en filpekare som heter fp
    // b) Öppna/Skapa filen "info" i binärt skrivläge
    if(fp == NULL)
        printf("Could not open file\n");
    else
    {
        // c) skriv hela arrayen numbers till filen (binärt)
    }

    // d) stäng filen

    return 0;
}
```

Svar:

```
a) FILE *fp;
b) fp = fopen("info", "wb");
c) fwrite(numbers, SIZE, sizeof(int), fp);
d) fclose(fp);
```

Exam - Programming

DVA117

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

School of Innovation, design and technology

Monday 2019-06-03

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]

Explain the difference between the following two expressions.

```
int number;
```

a) `number = 4`
b) `number == 4`

Answer: in a there is an assignment (number are set to 4) while in b a comparison is made (does number contain the data 4?)

Question 2 [2p]

What does it mean that

a) a variable is declared?
b) a variable is initialized?
c) a function is declared?
d) a function is defined?

Answer each question with a maximum of 4 sentences.

Answer:

a) Memory is allocated for the variable (the size of the memory is determined by the type) – the variable is created.
b) The variable is set to a starting value.
c) The head of the function is the function declaration – it states (to the compiler) the name of the function, what type is to be returned from the function and the number of and types of parameters that needs to be sent to the function.
d) The function definition is the implementation of the function – the code that makes the function to what it does.

Question 3 [1p]

What is the &-operator and *-operation (except for declaration) used for? To use an example in your explanation is suggested.

Answer: & is used to get the address where a variable is stored in memory. * is used to get the content at a specific address in memory.

Question 4 [1p]

What is missing in the calculation below for the result to be correct?

```
int totalSumOfPoints, nrOfStudents;
float averagePointsAtExam;
...
averagePointsAtExam = totalSumOfPoints/nrOfStudents;
```

Answer: A type conversion. Now a division is made between to integers, which will genereate a new integer (with the decimals cut).

Question 5 [4p]

a) In the program below – what is created on the heap and what is created on the stack? *(3p)*

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

int main(void) {

    int number = 5;
    float arrA[4] = {5, 10, 15, 20};
    int *pnumber = &number;
    float *arrB;
    arrB = (char*)malloc(5*sizeof(char));
    arrB[0] = 'a'; arrB[1] = 'b'; arrB[2] = 'c';
    return 0;
}
```

Write the following table on your answer paper and fill in if the variable/data is created/saved on the stack or on the heap.

	Stack or Heap?
The variable: `number`	Stack
The variable: `arrA`	Stack
The data: 5, 10, 15 & 20	Stack
The variable: `pnumber`	Stack
The variable: `arrB`	Stack
The data: 'a', 'b' & 'c'	Heap

b) For how long will memory allocated/created on the stac exist? And for how long will memory allocated/created on the heap exist? *(1p)*

Answer: Memory allocated on the stack exists in the block/scope where the variable is declared. Memory allocated on the heap exists until the the program frees it (using the function free( )).

Question 6 [4p]

Below you will find a short program where part om the program is missing (marked with a box). Each of the sub-tasks below (a-c) indicates one or more lines of code to be added where code is missing in the program.

Below you will also find a number of suggestions for printouts (1-6).

Your task is to pair each block of code (a-c) with the correct printout (1-6). There may be suggested printouts that are correct for several of the blocks of code and there are printouts that do not match any of the blocks.

```
#include <stdio.h>

int main(void) {

    int x = 0, y = 0;
    while(x < 5)
    {
        [          ]

        printf("%d%d  ", x, y);
        x = x + 1;
    }

    return 0;
}
```

Suggested printouts

1) 11 23 36 410

2) 13 34 53

3) 13 34 53 70

4) 11 34 59

5) 02 14 25 36 47

6) 01 12 24 36 48

a) (1p)

```
x = x + 1;
y = y + x;
```

b) (1p)

```
y = y + 2;
if(y > 4)
      y = y - 1;
```

c) (2p)

```
if(y < 5)
{
    x = x + 1;
    for(int i = 0; i < x; i++)
          y = y - 1;
}
y = y + 4;
```

Answer: a – 4 b – 5 c – 2

Question 7 [4p]

In this questions your should write a function named `countCharacters`, whose task is to count the number of characters in the string excluding the string ending character ('\0') – in other words, you should write you own strlen-function.

Start from the following program:

```
int main(void) {

    char input[100];
    int nrOfCharacters = 0;
    printf("Please enter a sentence: ");
    fgets(input, 100, stdin);

    /*Here, the function countCharactes should be called*/

    printf("The sentence consists of %d characters\n", nrOfCharacters);
    return 0;
}
```

The function should be called `countCharacters` and have void as return-type. What parameters the functions needs is up to you to decide. The task of the function is to count the numner of characters in a given string (in the program above the number of characters in the sting `input` should be counted). The '\0' character should not be included in the count. Note that the number of characters in the string should be printed in main. You are not allowed to use the strlen( )-function in your solution. *(3p)*

fgets() will also save the ending enter-character in the string input. It is up to you if the enter-character should be counted or not.

You should also write the function call to `countCharacters`, the call should be made where the comment is in the program above. *(1p)*

Solution:

Call: `countCharacters(input, &nrOfCharacters);`

```
void countCharacters(char *str, int *nr)
{
    int i;
    for(i = 0; str[i] != '\0'; i++);

    *nr = i;

}
```

Question 8 [2p]

Look at the following program

```
#include <stdio.h>
#define SIZE 5

int* update(int array[])
{
    for(int i = 0; i < SIZE; i++)
        array[i] = array[i] + 1;
    return array;
}

int main(void) {
    int arr[SIZE] = {1, 2, 3, 4, 5};
    arr = update(arr);

    for(int i = 0; i < SIZE; i++)
        printf("%d  ", *(arr+i));
    return 0;
}
```

Determine whether or not the program goes through the compiler.

If you think the compilation is successful – what will be the printout?
If you think the compilation fails – explain why it fails as well as how to solve the problem.

Answer: The compilation fils. At the function call to update the program tries to assign the array arr to the "new" data returned – an array is not assignable and an error will occure. It is enough to send the array to the function (update) – no return is needed.

Question 9 [12p]

In the following sub-tasks you will build parts of a program handling information about an aquarium.

a) Define your own type called `fish`, it should be able to contain the following information: *(1p)*

- `species` – for example ”Piraya” or ”Clownfisk”
- `age` – the age of the fish
- `teeth` – How many teeth does the fish have?

Answer:

```
struct fish
{
    char *species;
    int age;
    int teeth;
};
```

b) Declare a variable ”Snappy” of your own type created in sub-task a). Set the `species` to ”Piranha”, age to 4 and `teeth` to 69. *(2p)*

The function `strcpy(char *dst, char *src);` may be helpful.

Answer:

```
struct fish snappy;
strcpy(snappy.species, "Piranha");
snappy.age = 4;
snappy.teeth = 69;
```

c) Now declare a pointer ”`aquarium`” of your own type from sub-task a). Let the pointer point to a dynamically allocated memory large enough to hold information about 10 fishes. Also show how to test whether the allocation was successful or not. *(2p)*

Answer:

```
struct fish *aquarium = NULL;
aquarium = (struct fish*)calloc(10, sizeof(struct fish));
if(aquarium == NULL)
    printf("Error allocating");
else
{
    //Work with the memory
}
```

d) Write a function named `fillAquarium` that take (as parameter) a pointer to the aquarium (the dynamic memory allocated in sub-task c) as well as the number of spots in the aquarium. The function should let the user enter information about all the fishes (`species`, age and `teeth`). The return-type should be void.

Also write the function call to the function if the dynamic memory in sub-task c) is to be filled with information. *(3p)*

Answer:

```
void fillAquarium(struct fish *aq, int size)
{
     for(int i = 0; i < size; i++)
     {
          printf("Enter species: ");
          gets(aq[i].species);
          printf("Enter age: ");
          scanf("%d", &aq[i].age);
          printf("Enter number of teeth: ");
          scanf("%d", &aq[i].teeth);
      }
}
```

Call: `fillAquarium(aquarium, 10);`

e) Write a function called `addFishes` whose task is to increase the number of fishes that will fit in the aquarium – in other words, increase the amount of dynamically allocated memory. The new number of fishes must be sent to the function, other parameters and the return type of the functions is up to you to decide. If it is not possible to increase the size of the aquarium (the amount of memory), the old aquarium must be returned/retained.

Also show the funcition call if the array `aquarium` should be expanded to a total of 20 fishes. *(4p)*

Answer:

```
struct fish* addFishes(struct fish *aq, int newSize)
{
    struct fish* temp = (struct fish*)realloc(aq, newSize *
                                        sizeof(struct fish));
    if(temp != NULL)
         return temp;
    else
    {
         printf("Error");
         return aq;
    }
}
```

Call: `aquarium = addFishes(aquarium, 20);`

OR

```
void addFishes(struct fish **aq, int newSize)
{
     struct fish* temp = (struct fish*)realloc(*aq, newSize *
                                                sizeof(struct fish));
     if(temp != NULL)
         *aq = temp;
     else
         printf("Error");
}
```

Call: `addFishes(&aquarium, 20);`

Question 10 [3p]

In the program below there are 4 lines of code missing that handles an external file. Your task is to wirte the 4 lines.

```
#include <stdio.h>
#define SIZE 10

int main(void) {

    int numbers[SIZE];
    for(int i = 0; i < SIZE; i++)
    {
        printf("Enter number %d: ", i+1);
        scanf("%d", &numbers[i]);
    }

    // a) Declare a file-pointer called fp
    // b) Open/Create the file "info" for binary writing.
    if(fp == NULL)
        printf("Could not open file\n");
    else
    {
        // c) Write the entire array numbers to the file (binary).
    }

    // d) Close the file.

    return 0;
}
```

Answer:

```
a) FILE *fp;
b) fp = fopen("info", "wb");
c) fwrite(numbers, SIZE, sizeof(int), fp);
d) fclose(fp);
```