Tentamen - Programmering

DVA117

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

Akademin för innovation, design och teknik

Torsdag 2016-11-03

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

Skrivtid: 08.10 – 11.30
Hjälpmedel: Valfritt icke-elektroniskt material
Lärare: Caroline Uppsäll, 021-101456
(kan nås på telefon om du frågar tentavakten)

Preliminära betygsgränser

Betyg 3:	16p
Betyg 4:	23p
Betyg 5:	28p
Max:	**32p**

Allmänt

- All kod skall skrivas i standard ANSI C.
- Skriv tydligt vilken uppgift/deluppgift ditt svar anser.
- Skriv endast på bladets ena sida.
- Referera inte mellan olika svar.
- 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!
- Tips: Läs igenom hela tentamen innan du börjar skriva för att veta hur du ska disponera din tid.
-

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

Lycka till!
/Caroline

Uppgift 1 [1p]

Vad evaluerar följande logiska uttryck till – sant (1) eller falskt (0)?

```
int a = 0, x = 40, b = 6, y = 2, z = 4;

((a==x ||b != y+z) || !(b<x && a==1))

((0==40||6 != 2+4) || !(6<40&& 0==1))
(( 0   ||   0    ) || !( 1  &&   0 ))
(       0          || !      0      )
        0          || 1
```

RÄTT SVAR: 1 (sant)

Uppgift 2 [1p]

Antag följande egendefinierade typ och deklaration. Hur mycket minne allokeras för variabeln `myVolvo` på ett 32-bitars system?

```
struct car{
    char model[15];     //15 byte
    int productionYear; //4 byte
    float hp;           //4 byte
    char fuel;          //1 byte
};

struct car myVolvo; //RÄTT SVAR: 15+4+4+1 byte = 24 byte
```

Uppgift 3 [1p] (0,5p per sats)

Antag samma egendefinierade typ som i uppgift 2 ovan och följande deklaration

```
struct car *myBMW;
```

Skriv två olika satser som sätter värdet på `myBMW`'s `productionYear` till 2016.

```
myBMV->productionYear = 2016;
(*myBMW).productionYear = 2016;
```

Uppgift 4 [1p]

Om man skickar en array som ett argument till en funktion, vad är det då som kopieras/överförs till funktionen?

- värdet av elementen i arrayen
- första elementet i arrayen
- **adressen till första elementet i arrayen**
- adresserna till elementen i arrayen

Uppgift 5 [4p – 0,5p per deluppgift 1-8]

För var och en av tilldelningarna nedan ska du ange hur deklarationen av den efterfrågade variabeln måste se ut för att programmet ska kompilera och fungera som det är tänkt.

Antag följande egendefinierade typ.

```
struct time{
    int h,m,s;
};
```

Utgå från att följande deklarationer finns i programmet.

```
int x = 5, y = 10;
float n[10];
```

Exempel:

Tilldelning:	Ange deklarationen för	
`d = 5.0;`	`d`	svar: `float d;`

Förklaring: det som ska lagras i d vid tilldelningen är ett flyttal så därför måste d vara deklarerad som en flyttalsvariabel

Uppgift:

	Tilldelning:	Ange deklarationen för	
1)	`printf("%s", g);`	`g`	**`char *g; eller char g[20];`**
2)	`scanf("%c", &f);`	`f`	**`char f;`**
3)	`a.m = x;`	`a`	**`struct time a;`**
4)	`b->s = 58;`	`b`	**`struct time *b;`**
5)	`c = (float)y;`	`c`	**`float c;`**
6)	`r = *(n+3);`	`r`	**`float r;`**
7)	`i = &y;`	`i`	**`int *i;`**
8)	`printf("%.2f", e[4]);`	`e`	**`float e[5] //minst 5 element`**

Uppgift 6 [4p – 1p per deluppgift 1-4]

För följande funktionsanrop, ange funktionshuvudet (dvs det som används när man definierar och deklarerar funktionen).

Antag följande egendefinierade typ

```
struct time{
    int h,m,s;
};
```

Utgå från att följande deklarationer finns i programmet.

```
struct time now, *t;
int y, *u;
float arr[10];
```

Exempel:

Funktionsanrop:	Ange deklarationen för
`y = func(6.3);`	`func()` svar: `int func(float x);`

Förklaring: Eftersom func() tar ett flyttal som argument måste parametern vara av typen float (vad den heter är mindre viktigt).
Returvärdet tas emot i y som är deklarerad som en int, därför måste func() returnera en int.

Uppgift:

Funktionsanrop:	Ange deklarationen för
1) `int z = foo(arr, 5);`	`foo(`
`int foo(float a[10], int x); eller int foo(float *a, int x);`	
2) `now = compute(u, &y, 40);`	`compute()`
`struct time compute(int *x, int *y, int z);`	
3) `print_stuff("time", now.h);`	`print_stuff()`
`void print_stuff(char *str, int a);` **`ok med char str[5]; eller void* str)`**	
4) `t = init();`	`init()`
`struct time *init(void);`	

Uppgift 7 [3p]

Hur ser utskriften av följande kod ut? (3p)

```
#include <stdio.h>

int main()
{
    int i, num = 0, x = 0;

    for(i = 0; i <= 6; i++)
    {
        num = 0;
        while(num <= x)
        {
            printf("%d ", num);
            num++;
        }
        if (i < 3)
            x++;
        else
            x--;

        printf("\n");
    }

    return 0;
}
```

0
0 1
0 1 2
0 1 2 3
0 1 2
0 1
0

Uppgift 8 [5p] **(0,5p per hittat fel, 0,5p per korrekt lösning på fel – det finns totalt 6 fel så max är 6p och inte 5)**

Hur många fel finns det i koden (semantiska, syntaktiska, logiska)? Förklara samtliga fel du hittar samt hur satsen (raden kod) ska se ut för att fungera korrekt.

```
#include <stdio.h>

void letter_grade(int points, char *grade);

int main(void)
{
    int results[10] = {55, 34, 98, 76, 59, 62, 23, 87, 85, 49};
    char grades[10] = {};
    int close;

    for(i = 0; i <= 10; i++) //ej deklarerad. int i;
    {  //indexering utanför arrayens längd. i < 10 eller i <= 9 */
        close = letter_grade(results[i], &grades[i]);
        printf("\nStudent %d: %dp = %c", i+1, results[i], grades+i));
       /*grades+i är en adress, inte ett värde. Ska vara
         *(grades+i) eller grades[i]*/

        if(close = 1) //tilldelning ej jämförelse. if(close == 1)
            printf("\nStudent %d is close to a higher grade", i+1);
    }
    return 0;
}

/*fel returtyp, ska vara int*/
void letter_grade(int points, char *grade)
{
    if(points >= 0 && points < 50)
        *grade = 'F';
    else if(points >= 50 && points < 60)
        *grade = 'E';
    else if(points >= 60 && points < 70)
        *grade = 'D';
    else if(points >= 70 && points < 80)
        *grade = 'C';
    else if(points >= 80 && points < 90)
        *grade = 'B';
    else if(points >= 90 && points <= 100)
        *grade = 'A';
    else
        *grade = 'Z';
    if(points==89 || points==79 || points==69 || points==59 || points==49)
        return 1;
    else
        return 0;
}
```

Uppgift 9 [12p]

Utgå från följande typ som definierar låtar:

```
struct song
{
    char title[20];
    char artist[20];
    float length;
};
```

myAlbum.txt

```
3
Madonna
Like A Virgin
3.39
Madonna
Vogue
5.17
Madonna
Music
3.46
```

I deluppgift A, B och C nedan ska du skriva funktioner (alltså C-kod). Ingen av dessa funktioner får läsa något direkt från användaren (t.ex. scanf() eller gets()) eller skriva något på skärmen. Ingen av funktionerna kräver speciellt mycket kod så om du behöver skriva mer än en sida kod per funktion så gör du troligtvis någonting fel.

Deluppgift A (2p)

Skriv en funktion `createAlbum()` som tar ett antal som argument och returnerar en dynamiskt allokerad array av låtar som innehåller det antal platser som skickats in till funktionen. Om man t.ex. anropar `createAlbum(7)` ska alltså en dynamisk array med plats för 7 låtar av typen ovan returneras. Arrayen ska inte fyllas med någonting, funktionens syfte är enbart att allokera tillräckligt med minne.

```
struct song  *createAlbum(int size)
{
    return (struct song *)malloc(sizeof(struct song)*size);
}
```

Deluppgift B (4p)

En skivbolagsdirektör vill ha en funktion `canBeRecorded()` som avgör om en array av låtar kan spelas in på ett album. Hon vill att funktionen ska ta en dynamiskt allokerad array av låtar som argument, samt arrayens längd, och sedan returnera 1 (sant) eller 0 (falskt) beroende på om hela arrayen kan spelas in på ett album eller inte. Hon vil också att funktionen ska returnera hur långt (totalt antal minuter) albumet skulle bli om hela arrayen av låtar spelades in (går det inte att spela in albumet så kan längden -1 minuter returneras).

Hon har några regler för att en array av låtar ska få spelas in på ett album:

- Ingen låt får vara längre än 7 minuter
- Den totala speltiden av alla låtar tillsammans får inte överstiga 60 min
- Det får inte finnas fler än 12 låtar på ett album

Din uppgift är att skriva en funktionen beskriven ovan. Samtliga regler måste uppfyllas för att 1 (sant) ska få returneras från funktionen tillsammans med det totala antalet minuter.

```
int canBeRecorded(struct song *album, int size, float *totalLenght)
{
    int i;
    if(size > 12)
    {
        *totalLenght=-1;
        return 0;
    }
    *totalLenght = 0.0;
    for(i=0;i<size;i++)
    {
        if(album[i].length > 7.0)
        {
            *totalLenght = -1;
            return 0;
        }
        *totalLenght += album[i].length;
    }
    if(*totalLenght > 60)
    {
        *totalLenght = -1;
        return 0;
    }
    return 1;
}
```

Deluppgift C (6p)

Antag att det finns en textfil som heter myAlbum.txt. Skriv en funktion `readFromFile()` som tar in en filpekare till myAlbum.txt som argument och returnerar en dynamiskt allokerad array av låtar där varje låts information är hämtad från filen myAlbum.txt, Du kan anta att filen myAlbum.txt öppnas i läge "r" utan problem utanför funktionen. Funktionen ska läsa data från filen och fylla en dynamiskt allokerad array av låtar med datan. Funktionen ska givetvis anropa funktionen `createAlbum()` som du skapade i deluppgift A när minne ska allokeras. Har du inte löst deluppgift A så gör du anropet som du tror det ser ut baserat på beskrivningen av argument och returvärde i deluppgift A. Du ser hur filen myAlbum.txt ser ut högst upp på sidan, första raden i filen visar hur många låtar som finns beskrivna i filen.

```
struct song *readFromFile(FILE *fp)
{
    int size, i;
    struct song *album;
    fscanf(fp, "%d\n", &size); //newline
    album = createAlbum(size);
    if(album != NULL)
    {
        for(i = 0; i < size; i++)
        {
            fgets(album[i].artist,20, fp);
            fgets(album[i].title, 20, fp);
            fscanf(fp, "%f\n", &album[i].length); //newline
        }
    }

    return album;
}
```

Exam - Programming

DVA117

----------SOLUTIONS----------

School of innovation, design and technology

Thursday 2016-11-03

Write time: 08.10 – 11.30
Aids: Any non-electronic material
Teachers: Caroline Uppsäll, 021-101456
(can be reached by telephone if you ask the exam guard)

Preliminary grading limits
Grade 3: 16p
Grade 4: 23p
Grade 5: 28p
Max: **32p**

Generally

- All code should be written in standard ANSI C.
- Write clearly what task/sup-task your answer consider.
- Do only use one side of the paper.
- Do not refer between answers.
- If you are unsure of a meaning of a question, write down your assumption.
- *Unreadable/incomprehensible answers will not be marked.*
- Comment your code!
- Hint: To know how to allocate your time, read through the entire exam before you start writing.

----------SOLUTIONS----------

Good luck!
/Caroline

Question 1 [1p]

What does the following logical expression evaluate to – true (1) or false (0)?

```
int a = 0, x = 40, b = 6, y = 2, z = 4;

((a==x ||b != y+z) || !(b<x && a==1))

((0==40||6 != 2+4) || !(6<40&& 0==1))
(( 0   ||   0    ) || !( 1  &&   0 ))
(       0          || !      0      )
        0          || 1
```

CORRECT ANSWER: 1 (true)

Question 2 [1p]

Assume the following defined type and declaration. How much memory is allocated for the variable `myVolvo` on a 32-bit system?

```
struct car{
    char model[15];     //15 byte
    int productionYear; //4 byte
    float hp;           //4 byte
    char fuel;          //1 byte
};

struct car myVolvo; //ANSWER: 15+4+4+1 byte = 24 byte
```

Question 3 [1p] (0,5p/expression)

Assume the same defined type as in question 2 above, and the following declaration

```
struct car *myBMW;
```

Write two different expressions to set the value of `myBMW`'s `productionYear` to 2016.

```
myBMV->productionYear = 2016;
(*myBMW).productionYear = 2016;
```

Question 4 [1p]

If you send an array as an argument to a function, what is copied/transferred to the function?

- The values of the elements in the array
- The first element in the array
- **The address of the first element in the array**
- The addresses of the elements in the array

Question 5 [4p – 0,5p/sub-task 1-8]

For each of the assignments below, please indicate how the declaration of the requested variable must look like for the program to compile and work as intended.

Assume the following defined type.

```
struct time{
    int h,m,s;
};
```

Assume that the following declarations exist in the program.

```
int x = 5, y = 10;
float n[10];
```

Example:

Assignment:	Write declaration for:	
`d = 5.0;`	`d`	answer: `float d;`

Explanation: a floating-point number is assigned to d, d must be a float-type variable

Task:

	Assignment:	Write declaration for:	
9)	`printf("%s", g);`	`g`	**`char *g; or char g[20];`**
10)	`scanf("%c", &f);`	`f`	**`char f;`**
11)	`a.m = x;`	`a`	**`struct time a;`**
12)	`b->s = 58;`	`b`	**`struct time *b;`**
13)	`c = (float)y;`	`c`	**`float c;`**
14)	`r = *(n+3);`	`r`	**`float r;`**
15)	`i = &y;`	`i`	**`int *i;`**
16)	`printf("%.2f", e[4]);`	`e`	**`float e[5] //at least 5`**

Question 6 [4p – 1p/sub-task 1-4]

For each of the function calls below please indicate how the function declaration must look like for the call to work.

Assume the following defined type.

```
struct time{
    int h,m,s;
};
```

Assume that the following declarations exist in the program.

```
struct time now, *t;
int y, *u;
float arr[10];
```

Example:

Function call:	Write declaration for:
`y = func(6.3);`	`func()`

answer: `int func(float x);`

Explanation: Since func() takes a floating-point as argument the parameter must be of type float (what it is called is not as important).
The return value is copied to y which is declared as an integer variable, therefore func() must return an integer.

Task:

Function call:	Write declaration for:
5) `int z = foo(arr, 5);`	`foo(`

`int foo(float a[10], int x); or int foo(float *a, int x);`

6) `now = compute(u, &y, 40);`	`compute()`

`struct time compute(int *x, int *y, int z);`

7) `print_stuff("time", now.h);`	`print_stuff()`

`void print_stuff(char *str, int a);`
`ok with char str[5] or void* str`

8) `t = init();`	`init()`

`struct time *init(void);`

Question 7 [3p]

What does the output of the following program look like?

```
#include <stdio.h>

int main()
{
    int i, num = 0, x = 0;

    for(i = 0; i <= 6; i++)
    {
        num = 0;
        while(num <= x)
        {
            printf("%d  ", num);
            num++;
        }
        if (i < 3)
            x++;
        else
            x--;

        printf("\n");
    }

    return 0;
}
```

0
0 1
0 1 2
0 1 2 3
0 1 2
0 1
0

Question 8 [5p] **(0,5p/correct error, 0,5p/correct correction of error)**

How many errors (syntactic, semantic, logic) can you find in the program below? Explain every error you can find and show (in code) how it should look like when corrected.

```
#include <stdio.h>

void letter_grade(int points, char *grade);

int main(void)
{
    int results[10] = {55, 34, 98, 76, 59, 62, 23, 87, 85, 49};
    char grades[10] = {};
    int close;

    for(i = 0; i <= 10; i++) //ej deklarerad. int i;
    {  //indexering utanför arrayens längd. i < 10 eller i <= 9 */
        close = letter_grade(results[i], &grades[i]);
        printf("\nStudent %d: %dp = %c", i+1, results[i], grades+i));
       /*grades+i är en adress, inte ett värde. Ska vara
         *(grades+i) eller grades[i]*/

        if(close = 1) //tilldelning ej jämförelse. if(close == 1)
            printf("\nStudent %d is close to a higher grade", i+1);
    }
    return 0;
}

/*fel returtyp, ska vara int*/
void letter_grade(int points, char *grade)
{
    if(points >= 0 && points < 50)
        *grade = 'F';
    else if(points >= 50 && points < 60)
        *grade = 'E';
    else if(points >= 60 && points < 70)
        *grade = 'D';
    else if(points >= 70 && points < 80)
        *grade = 'C';
    else if(points >= 80 && points < 90)
        *grade = 'B';
    else if(points >= 90 && points <= 100)
        *grade = 'A';
    else
        *grade = 'Z';
    if(points==89 || points==79 || points==69 || points==59 || points==49)
        return 1;
    else
        return 0;
}
```

Question 9 [12p]

Assume the following type defining songs:

```
struct song
{
    char title[20];
    char artist[20];
    float length;
};
```

myAlbum.txt

```
3
Madonna
Like A Virgin
3.39
Madonna
Vogue
5.17
Madonna
Music
3.46
```

In sub-task A, B and C below you should write functions (C-code). None of these functions may take input directly from the user (e.g. scanf(), gets()) or print anything on the screen. None of the functions require very much code so if you need to write more than one page of code per function (sub-task) you most likely are doing something wrong.

Sub-task A (2p)

Write a function `createAlbum()` that takes a number as argument and returns a dynamically allocated array of songs that contain the number of songs given to the function. E.g. if the function is called with `createAlbum(7)`, a dynamically allocated array of 7 songs (as defined above) should be returned from the function. The array should not be filled with any data, the functions purpose is merely to allocate enough memory.

```
struct song  *createAlbum(int size)
{
    return (struct song *)malloc(sizeof(struct song)*size);
}
```

Sub-task B (4p)

A director for a record company wants a function `canBeRecorded()` that determines whether an array of songs can be recorded on an album or not. She wants the function to take an dynamically allocated array of songs as argument as well as the length of the array and then return 1(true) or 0(false) depending on whether the entire array of songs can be recorded on an album or not. She also wants the function to return the total length (minutes) of the album if recorded. If the album cannot be recorded the function should return -1 minutes.

She has a few rules that must be fulfilled for an array of songs to be able to be recorded on an album:

- No single song may be longer then 7 minutes
- The total playing time of all tracks together may not exceed 60 minutes
- There cannot be more then 12 songs on an album.

Your task us the write the function `canBeRecorded()` described above. All rules must be fulfilled for the function to return 1 (true) and the total playing time in minutes.

```
int canBeRecorded(struct song *album, int size, float *totalLenght)
{
    int i;
    if(size > 12)
    {
        *totalLenght=-1;
        return 0;
    }
    *totalLenght = 0.0;
    for(i=0;i<size;i++)
    {
        if(album[i].length > 7.0)
        {
            *totalLenght = -1;
            return 0;
        }
        *totalLenght += album[i].length;
    }
    if(*totalLenght > 60)
    {
        *totalLenght = -1;
        return 0;
    }
    return 1;
}
```

Sub-task C (6p)

Suppose there is a text file named myAlbum.txt. Write a function `readFromFile()` that takes a file pointer to myAlbum.txt as argument and returns a dynamically allocated array of songs in which each songs information is retrieved from the file myAlbum.txt. You can assume that myAlbum.txt is successfully opened in r-mode outside of the function. For allocating memory the function should call `createAlbum()` from sub-task A. If you have not solved sub-task A you should still be able to do the function call as you think it should look like based on the description given. You can see how the file myAlbum.txt looks like at the top of the page, the first row in the file describes how many songs are stored in the file.

```
struct song *readFromFile(FILE *fp)
{
    int size, i;
    struct song *album;
    fscanf(fp, "%d\n", &size); //newline
    album = createAlbum(size);
    if(album != NULL)
    {
        for(i = 0; i < size; i++)
        {
            fgets(album[i].artist,20, fp);
            fgets(album[i].title, 20, fp);
            fscanf(fp, "%f\n", &album[i].length); //newline
        }
    }

    return album;
}
```