Tentamen - Programmering

DVA117

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

Akademin för innovation, design och teknik

Torsdag 2019-01-17

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: ~~16p~~ 15p (reviderad betygsgräns)
Betyg 4: 24p
Betyg 5: 28p
Max: 32p

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 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!
- 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.

Lycka till!
/Caroline

----------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 med maximalt 3 meninga vad syntax är (i samband med programmeringsspråk).

Förklaring som till viss del är rätt och till viss del är fel ger inga poäng. Otillräcklig förklaring/beskrivning ger inga poäng.

Svar: Syntax är den uppsättning grammatiska regler som gäller för programmeringsspråket.

Uppgift 2 [2p]

Beskriv med maximalt 5 meninga vad som är skillnaden på en global och en lokal variabel. Beskrivningen ska vara gjord på ett sådant sätt så att den i sig beskriver vad en global samt en lokal variabel är. Ge gärna ett exempel.

Förklaring som till viss del är rätt och till viss del är fel ger inga poäng. Otillräcklig förklaring/beskrivning ger inga poäng.

Svar: En global variabel deklareras utanför själva programmet (utanför funktionerna) och existerar genom hela programmet. En lokal variabel existerar endast i det block (t.ex. funktion) där den blivit deklarerad.

Uppgift 3 [1p]

Vad evaluerar följande uttryck till (sant/1 eller falskt/0) med följande variabeldeklarationer:

```
int a = 9, b = 4, x = 2, y = 13;

(a % 2 == 0 || a/b <= x) && !(y - a < b)
```

Svar: 1/Sant

```
(9 % 2 == 0 || 9/4 <= 2) && !(13 - 9 < 4)
(  1   == 0 ||  2  <= 2) && !(   4   < 4)
(     0     ||     1   ) && !(     0    )
(           1           ) && 1
                          1
```

Uppgift 4 [5p]

Vi ska skriva ett program som frågar efter två tal, och sen skriver ut resultatet av om man tar det ena talet upphöjt till det andra. Körexempel, med användarens inmatning understruken:

```
Ange basen: 2
Ange exponenten: 6
Resultat: 64
```

Programmet ska alltså beräkna 2^6, vilket också kan skrivas som 2*2*2*2*2*2.

Så här blev programmet:

```
/* Ett program som räknar ut ett tal upphöjt till ett annat */

#include <stdio.h>;

int power(int base, int exponent) {
    int result = 1;
    for (i = 0; i < base; i--)
        result = result * base;
}

int main(void) {
    int theBase; theExponent;
    printf("Enter base: ");
    scanf(&theBasen);
    printf("Enter exponent: ");
    scanf("%c", &theExponenten);
    power(theBase, theExponent);
    int theResult;
    theResult = power;
    printf("Result: ", theResult);
    return 0;
}
```

Tyvärr har det blivit ganska många fel i programmet. Tala om vilka fel det är, och visa vad det borde stå i stället!

Programmet finns även på sista sidan i tentan. Du kan göra rättelserna på den sidan, och sedan riva loss den och lämna in den tillsammans med dina övriga lösningar, så slipper du skriva av programkoden.

Svar:

```
#include <stdio.h>;

int power(int base, int exponent) {
    int result = 1;
    int i;
    for (i = 0; i < exponent; i++)
        result = result * base;
    return result;
}

int main(void) {
    int theBase, theExponent;
    printf("Enter base: ");
    scanf("%d", &theBasen);
    printf("Enter exponent: ");
    scanf("%d", &theExponenten);
    int theResult = power(theBase, theExponent); //detta eller de två
raderna nedan
    int theResult;
    theResult = power(basen, exponenten);
    printf("Result: %d ", theResult);
    return 0;
}
```

Totalt 10 fel

0.5p per fel.

Uppgift 5 [4p]

Vad skrivs ut av nedanstående program?

```
#include <stdio.h>

int main(void)
{
    int a = 1, b = 2, i = 3;
    int *p = &b;
    while (i < 10) {
        a = *p;
        if (i < 4 || i == 8)
        {
            i += 3;
            p = &a;
        }
        else
        {
            i++;
            p = &b;
        }
        *p = *p + 1;
    }
    printf("a = %d\nb = %d\ni = %d\n", a, b, i);
    return 0;
}
```

Svar:

a = 5 (1.5p)
b = 4 (1.5p)
i = 11 (1p)

Uppgift 6 [4p]

Din uppgift är att skriva en program som ber användaren om en sträng och sedan skriver ut alla siffror (0-9) som existerar i strängen. Du kan begränsa strängens maximala längd till 50. Om strängen inte innehåller några siffror så kan utskriften lämnas tom. Tänk på att strängen ska kunna innehålla mellanslag.

Ex:

```
Ange en sträng: pro0gram7mer5ing9 ar 1roligt2!
0 7 5 9 1 2
```

Tips: Funktionen isdigit som finns i biblioteket ctype.h kan användas för att avgöra om ett tecken är en siffra eller inte. Om tecknet som skickas in till funktionen inte är en siffra returneras 0 (*falskt*), annars returneras någonting skillt från 0 (med andra ord *sant*).

Svar:

```
#include <stdio.h>
#include <ctype.h>

int main(void)
{
    char str[51];
```

```
    printf("Enter a string: ");
    gets(str);

    for(int i = 0; str[i] != '\0'; i++)
    {
        if(isdigit(str[i]))
            printf("%c ", str[i]);
    }
    return 0;
}
```

Uppgift 7 [15p]

Din uppgift i nedsanstående deluppgifter är att göra (i alla fall delar av) ett spel där man går runt i en grotta (bestående av olika rum) och träffar på olika monster (spelidén går fortfarande att utveckla…).

Ett monster har ett unikt nummer, ett namn och en storlek. Storleken är ett flyttal, till exempel 10.2 och namnet kan innehålla högst tio tecken (t.ex. Cute-Wumps) – glöm inte att namnet också måste ha plats för strängsslutstecknet.

Varje rum i grottan har ett unikt nummer och det innehåller upp till fem monster.

a) *[1p]* Definiera en posttyp som heter **struct Monster** och som innehåller data om ett monster enligt beskrivningen ovan.

```
struct Monster
{
    int monsterNr;
    char name[11];
    float size;
};
```

b) *[1p]* Deklarera en variabel av typen struct Monster och initiera den med följande data:
nummer: 17
namn: Gluggo
storlek: 2000.0

```
struct Monster myMonster;
myMonster.monsterNr = 17;
strcpy(myMonster.name, "Gluggo");
myMonster.size = 2000.0;
```

c) *[1p]* Skapa en posttyp som heter **struct Room** och som innehåller data om ett rum. Varje rum i spelet ska ha ett unikt nummer samt en array av de högst fem monster som befinner sig i rummet och en variabel som talar om hur många monster som befinner sig i rummet.

```
struct Room
{
      int roomNr;
      struct Monster monstersInRoom[5];
      int nrOfMonstersInRoom;
};
```

d) *[2p]* Spelet kan hanera högst 100 rum, så skapa en dynamiskt allokerad array som heter **rooms** och som har plats för 100 poster av **structen Room**. Skapa också en variabel som

heter **numberOfRooms** som håller reda på hur många platser i arrayen som för tillfället används.

```
struct Room *game =(struct Room*)malloc(100 * sizeof(struct Room));
int numberOfRooms;
```

e) *[4p]* Skriv funktionen **totalAmountOfMonsters**. Man ska kunna anropa funktionen med arrayen **rooms** och variabeln **numerOfRooms** som parameter och så ska funktionen returnera det totala antalet monster som finns i alla rummen. Funktionen **får inte** göra några inmatningar eller utskrifter. Returtyp på funktionen **måste** vara void.

Du ska förutom att skriva själva funktionen som beskrivs ovan även visa hur anropet till funktionen ser ut samt hur utskriften av resultatet ser ut. Du kan här anta att det finns ett antal rum med data i spelet samt att variabeln numnberOfRooms är korrekt uppdaterad till hur många rum som faktiskt existerar i spelet.

```
void totalAmountOfMonsters(struct Room *game, int nrOfRooms,
                           int *totalNrOfMonsters)
{
    for(int i = 0; i < nrOfRooms; i++)
    {
        *totalNrOfMonsters = *totalNrOfMonsters +
                             game[i].nrOfMonstersInRoom;
    }
}

//ANROP:
int monstersTotal = 0;
totalAmountOfMonsters(rooms, numberOfRooms, &monstersTotal);
printf("%d/n", monstersTotal);
```

f) *[6p]* Skriv funktionen **biggestMonsterInRoom**. Funktionens uppgift är att returnera det största monstret i ett angivet rum. Funktionen får inte hantera någon in- eller utmatning.

Du ska använda följande funktionshuvud till funktionen:

```
struct Monster biggestMonsterInRoom(struct Room *game, int nrOfRooms,
                                    int roomNr);
```

Där game är den dynamiska arrayen av rum, nrOfRooms är antalet rum som existerar i spelet och roomNr är det rum där funktionen ska söka fram det största monstret (rummets unika nummer).

Du kan i funktionen anta följande:

- roomNr är ett rum som existerar i game
- det aktuella rummet innehåller minst ett monster
- allt data för rummet samt det/de monster som finns i rummet är satt till korrekt data

Du ska förutom själva funktionen också visa hur anropet till funktionen ser ut. Du ska hitta det största monstret i rummet med det unika numret 27 och du kan arbeta med den dynamiska arrayen av rum samt variabeln numberOfRooms som du skapade i uppgift d) ovan. Du kan anta att dessa består av korrekt inlagd data samt att numberOfRooms är koorrekt uppdaterad.

Lösning:

```
struct Monster biggestMonsterInRoom(struct Room *game, int nrOfRooms,
                                    int roomNr)
{
    struct Monster biggest;
    for(int i = 0; i < nrOfRooms; i++)
    {
        if(game[i].roomNr == roomNr)
        {
            biggest = game[i].monstersInRoom[0];
            for(int j = 1; j < game[i].nrOfMonstersInRoom; j++)
            {
                if(biggest.size < game[i].monstersInRoom[j].size)
                    biggest = game[i].monstersInRoom[j];
            }
        }
    }
    return biggest;
}
```

```
//ANROP:
 struct Monster myMonster;
 myMonster = biggestMonsterInRoom(rooms, numberOfRooms, 4);
```

Exam - Programming

DVA117

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

School of Innovation, design and technology

Thursday 2019-01-17

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: ~~16p~~ 15p
Grade 4: 24p
Grade 5: 28p
Max: **32p**

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 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!
- 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.

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

Good luck!
/Caroline

Question 1 [1p]

Explain with a maximum of 3 sentences what syntax is (when talking about programming langugages).

Explanation that is to some extent correct and to some extent wrong does not give any points. Insufficient explanations/descriptions gives no points.

Answer: Syntax is the set of grammatical rules that apply to the programming language.

Question 2 [2p]

Describe with at maximum of 5 sentences what the difference is between a global and a local variable. The description must be made in such a way that it describes in itself what a global and a local variable is. It is ok to make an example as complement to the description.

Explanation that is to some extent correct and to some extent wrong does not give any points. Insufficient explanations/descriptions gives no points.

Answer: a global variable is declared outside of any function in the program and exists throughout the entire program. A local variable exists only in the block/scope (for example function) where it has been declared.

Question 3 [1p]

What does the following expression evaluate to? (true/1 or false/0).

```
int a = 9, b = 4, x = 2, y = 13;

(a % 2 == 0 || a/b <= x) && !(y - a < b)
```

Answer: 1/True

```
(9 % 2 == 0 || 9/4 <= 2) && !(13 - 9 < 4)
(  1   == 0 ||  2  <= 2) && !(   4   < 4)
(     0     ||     1   ) && !(     0    )
(           1           ) && 1
                          1
```

Question 4 [5p]

We have a task to write a program that asks the user for two integer numbers and then writes the result calculated should be the the first number rased to the second number. Se example with user input underlined:

```
Enter base: 2
Enter exponent: 6
Result: 64
```

With this input the program should calculate $2^{6,}$ which also can be written as 2*2*2*2*2*2.

This is how the program turned out:

```
/* A program that calculates a number raised to another number */

#include <stdio.h>;
```

```
int power(int base, int exponent) {
    int result = 1;
    for (i = 0; i < base; i--)
        result = result * base;
}

int main(void) {
    int theBase; theExponent;
    printf("Enter base: ");
    scanf(&theBasen);
    printf("Enter exponent: ");
    scanf("%c", &theExponenten);
    power(theBase, theExponent);
    int theResult;
    theResult = power;
    printf("Result: ", theResult);
    return 0;
}
```

Unfortunately there have been quite a few errors in the program. Your task is to find the errros made and suggest how they can be corrected.

The program is also available on the last page of the exam. If you like you can make the corrections on that page, tear it off and submit it along with your other solutions.

Answer – see the Swedish question.
There is a totalt of 10 errors – each correctly corrected error gives 0.5p.

Question 5 [4p]

What is the output of the following program?

```
#include <stdio.h>

int main(void)
{
    int a = 1, b = 2, i = 3;
    int *p = &b;
    while (i < 10) {
        a = *p;
        if (i < 4 || i == 8)
        {
            i += 3;
            p = &a;
        }
        else
        {
            i++;
            p = &b;
        }
        *p = *p + 1;
    }
    printf("a = %d\nb = %d\ni = %d\n", a, b, i);
    return 0;
}
```

Answer:
a = 5 (1.5p)

```
b = 4 (1.5p)
i = 11 (1p)
```

Question 6 [4p]

Your task is to write a program that asks the user for a string and then prints all the digits (0-9) that exists in the string. You can limit the maximum length of the string to 50. It the string does not contain any digits you can leave the output blanc. Keep in mind that the string should be able to contain spaces.

Example:

```
Enter a string: pro0gram7m5ing9 is 1fun2!
0 7 5 9 1 2
```

Tip: The function isdigit that can be found the library ctype.h can be used to determine if a characer is a digit or not. If the character sent to the function isn't a digit (0-9) the function will return 0 (*false*), if it indeed is a digit the function returns something different from zero (meaning *true*)

Answer – see the Swedish question

Question 7 [15p]

For Answers – see the Swedish question

Your task in the following subtasks is the create (at least parts of) a game in which you walk around in a cave (consisting of different rooms) and meet different monsters (the game idea is still under development…).

Each monster has a unique number (ID), a name and a size. The size is a floating point number, for example 10.2 and the name can contain a maximum of ten characters (eg. Cute-Wumps) – don't forget the '\0' in the name.

Each room in the cave will have a unique numner (ID) and it will contain up to five monsters.

a) *[1p]* Define a struct type named **struct Monster** and that contains all the data about a monster described above.

b) *[1p]* Declare a variable of the new type struct Monster and initialize it with the following data:
unique number: 17
name: Gluggo
size: 2000.0

c) *[1p]* Create a new struct type named **struct Room** that contains data about a room in the cave. Each room in the gave should have a unique number (ID) and contain an array of maximum five monsters (the monsters in the room) as well as a variable that keeps track of how many monsters there are in the room at any given point.

d) *[2p]* The gave can handle at the most 100 rooms. Create a dynamically allocated array named **rooms** that can store up to 100 entities of the struct **Room**. Do also declare a variable named **numberOfRooms** that keeps track of how many elements of the dynamic array that are in use.

e) *[4p]* Write the function **totalAmountOfMonsters**. One should be able to call the function with the array **rooms** and the variable **numberOfRooms** as parameters, the function should then return the total amount of monsters in all the rooms combined. The function is not allowed to make any input or output. **The return type of the function must be void.**

You should also show how the call to the function will look like as well as the output of the information (in code). You can assume that there are a number of rooms created in the game and that the variable **numberOfRooms** is up to date.

f) *[6p]* write the function **biggestMonsterInRoom**. The function should return the biggest monster in a given room. The function is not allowed to take any input directly from the user or make any output.

You must use the following function declaration:

```
struct Monster biggestMonsterInRoom(struct Room *game, int nrOfRooms,
                                    int roomNr);
```

were game is the dynamically allocated array of rooms, nrOfRooms is the number of rooms that exists in the game and roomNr is the unique room number in which the function should find the biggest monster.

You can assume the following when creating the function:

- roomNr is an existing room number in the game.
- The room with roomNr contains the minimum of one monster.
- All the data for the rooms as well as the monster/monsters in the rooms are set correctly

You should also show what the call to the function looks like. With the call you should find the biggest monster in the room with number 27 and you can work with the dynamic array and numberOfRooms variable that you created in subtask d). You can assume that these are updated with data in a correct way.