FACIT!!

Tentamen - Programmering

DVA117

Akademin för innovation, design och teknik

Tisdagen 2014-11-06

Skrivtid: 08.10 - 11.30
Hjälpmedel: Inga
Lärare: Caroline Uppsäll, 021-101456
Robert Suurna, 021-151790

Preliminära betygsgränser

Betyg 3: p
Betyg 4: p
Betyg 5: p
Max: p

Bonuspoäng

Endast bonuspoäng som intjänats under kursinstansen HT13, p1, kan användas på den här tentamen. Bonuspoängen uppgår till max 6 poäng.

Allmänt

- All kod skall skrivas i standard ANSI C.
- Påbörja varje ny uppgift på nytt blad och skriv bara på ena sidan av pappret. Deluppgifter (a, b, c,…) kan skrivas på samma blad
- 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 tentan innan du börjar skriva för att veta hur du ska disponera din tid.

Lycka till!

FACIT!!

Prioritet och associativitet hos operatorerna i C

Högsta prioritet högst upp i tabellen, lägsta prioritet längst ner i tabellen. Grupperade operatorer har samma prioritet.

Table A.5 Summary of C Operators (Programming in C, Stephen G. kochan)

Operator	Description	Associativity
`()`	Function call	
`[]`	Array element reference	
`->`	Pointer to structure member reference	Left to right
`.`	Structure member reference	
`-`	Unary minus	
`+`	Unary plus	
`++`	Increment	
`--`	Decrement	
`!`	Logical negation	
`~`	Ones complement	Right to left
`*`	Pointer reference (indirection)	
`&`	Address	
`sizeof`	Size of an object	
`(type)`	Type cast (conversion)	
`*`	Multiplication	
`/`	Division	Left to right
`%`	Modulus	
`+`	Addition	Left to right
`-`	Subtraction	
`<<`	Left shift	Left to right
`>>`	Right shift	
`<`	Less than	
`<=`	Less than or equal to	Left to right
`>`	Greater than	
`=>`	Greater than or equal to	
`==`	Equality	Left to right
`!=`	Inequality	
`&`	Bitwise AND	Left to right
`^`	Bitwise XOR	Left to right
`|`	Bitwise OR	Left to right
`&&`	Logical AND	Left to right
`||`	Logical OR	Left to right
`?:`	Conditional	Right to left
`= *= /= %= += -= &= ^= |= <<= >>=`	Assignment operators	Right to left
`,`	Comma operator	Right to left

Uppgift 1 [7p]

För varje deluppgift kan ett eller flera svar vara korrekta. Varje deluppgift är värd 1p. Anges fel svarsalternativ ger det avdrag. Lägsta poäng per deluppgift är 0p. Svara genom att ange bokstaven för deluppgiften och den eller de siffror som motsvarar de rätta svarsalternativen
(ex1: H2, ex2: H3,4)

A. Vilken/vilka av följande alternative är deklarationer? A1,3

1. *`static int x;`*
2. `float square (float x) { ... }`
3. *`double pow(double, double);`*

B. Vilken/vilka av följande typer kan användas i en switch-case-sats? B1,2,4

1. *Character*
2. *Integer*
3. Float
4. *Enum*

Om bara ett rätt svar = 0p

C. Vad kallar vi följande rader kod i ett program? C2

```
struct point{
   int x,y;
};
```

1. Deklaration
2. *Definition*
3. Funktion
4. Deklaration och Definition

D. En pekare är? D3

1. Ett nyckelord som används för att skapa variabler
2. En variabel som lagrar adressen för en instruktion
3. *En variabel som lagrar adressen för en annan variabel*
4. Alla ovanstående

E. Om en variabel är en pekare till en post (struct), vilken/vilka av följande operator/operander används för att komma åt ett fält i posten via pekarvariabeln? E2,3

1. .
2. ->
3. ** och .*
4. *

Om bara två svar och det ena är fel = -0,5p

F. Om man skickar en array som ett argument till en funktion, vad är det då som överförs till funktionen? F3

1. Värdet av element i arrayen
2. Första element i arrayen
3. *Basadress av arrayen*
4. Hela arrayens innehåll

G. Vilket/vilka påståenden är korrekta för begreppet ”Socket”? G1,4

1. *Socket är en databuffert som delas mellan operativsystemet och en process*
2. Uppkopplingsfasen mellan två datorer som använder sockets är symmetrisk
3. Minst möjliga överföring mellan två enheter som använder sockets är 1 bit
4. *Socket gör det möjligt att skriva generell kod för datorkommunikation*

Uppgift 2 [3p]

A. Vad kommer följande program att skriva ut?

```
#include<stdio.h>

int main()
{
    int i, a[] = {2, 4, 6, 8, 10};
    change(a, 5);
    for(i=0; i<=4; i++)
        printf("%d, ", a[i]);
    return 0;
}

void change(int *b, int n)
{
    int i;
    for(i=0; i<n; i++)
        *(b+1) = *(b+i)+5;
}
```

Svar (2p): 2, 15, 6, 8, 10

B. Vid kompilering av ovanstående kod ger kompilatorn en varning, varför?
Svar (1p): Funktionen change() är inte deklarerad i början av programmet (implicit declarations of funktion)

Uppgift 3 [3p]

Beskriv steg för steg hur man går till väga för att i ett c-program öppna en fil i binärformat och skriva en hel post (struct) till filen. Du ska förutsätta att posten är definierad och deklarerad i en variabel som heter **`bokInfo`**. Beskriv tillvägagångssättet med dina egna ord som en punktlista i rätt ordning och tänk på att ta med alla delar som krävs.

Svar:

- Deklarera en filpekare
- Anropa funktionen fopen med flaggorna wb
- Skriva till filen genom att använda funktionen write med filpekaren och variabeln bokInfo
- Stänga filen med funktionen fclose och filpekaren

Uppgift 4 [10p]

A. Skapa en datatyp som ska innehålla information om en persons personnummer. Datatypen ska hålla födelseår, födelsemånad, födelsedag samt personens fyra sista siffor. Datatypen ska alltså hålla fyra variabler.

Lösningsförslag:

```
typedef struct{
        int year, month, day, lastFourDigits;
   }personarNr;
```

B. Skapa en datatyp som ska hålla information om en anställd på ett företag. Den information som ska anges är namn (förnamn och efternamn), personnummer (här ska datatypen från deluppgift A användas) och aktuell årslön.

```
Lösningsförslag:
typedef struct{
    char firstName[20], lastName[20];
    personalNr pNr;
    float yearlySalary;
}employee;
```

C. Skapa en funktion som frågar användaren efter hur många anställda som finns i företaget, skapa sedan en array av lika många anställda (använd dynamiskt minne och datatyperna från tidigare deluppgifter ovan). Glöm inte felhantering.

```
Lösningsförslag:
employee* allocateMemory()
{
   int size;
   employee *arr;
   printf("How many employees: ");
   scanf("%d", &size);
   arr = (employee *)malloc(sizeof(employee)*size);
   if(arr == NULL)
          printf("could not allocate memory");
  else
         printf("memory allocated");
  return arr;
}
```

D. Skapa nu två funktioner (`search` och `update`). `search` är den första funktionen som anropas, här ska användaren kunna söka efter en anställd baserat på efternamn. Om den anställde hittas ska funktionen `update` anropas med en pekare till den aktuella posten som inparameter. Funktionen `update` ska låta användaren uppdatera årslönen för den anställda. Funktionerna ska använda det ni skapat i tidigare deluppgifter)

Tips: för att jämföra två strängar kan du använda dig av funktionen
`int strcmp(const char * str1, const char * str2);`
som returnerar 0 om strängarna är lika och någonting skilt från 0 om de inte är lika.

```
Lösningsförslag:
void update(employee *personToupdate)
{
    printf("New salary: ");
    scanf("%f", &(personToupdate->yearlySalary));
}

void search(employee *arr, int size)
{
    char searchFor[20];
    int i;
    printf("Name to search for: ");
    gets(searchFor);
    for(i = 0; i < size; i++)
        if((strcmp((arr+i)->lastName, searchFor) == 0)
           update(arr+i);
}
```