FACIT!!

Tentamen - Programmering

DVA117

Akademin för innovation, design och teknik

Tisdagen 2013-11-05

Skrivtid: 14.10 – 17.30
Hjälpmedel: Inga
Lärare: Caroline Uppsäll, 021-101456
Robert Suurna, 021-151790

Preliminära betygsgränser

Betyg 3:	25p
Betyg 4:	34p
Betyg 5:	42p
Max:	45p

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]

a) Vad innebär det att ett program kompileras?
(1p): Den kod som man skrivit översätts till maskinkod som kan tolkas av en dator (skapar en exekverbar fil)

b) Beskriv i ord och bild vad fält (en array) är och varför man använder fält i sina program?
(2p): Array är en hel följd av variabler (värden), på rad i minnet, som var och en kan nås genom att ange värdet på ett (eller flera) index.
Ett sätt att gruppera data som är av samma typ under samma namn (minska ner antal variabelnamn, lättare hantering).

c) Beskriv skillnaden mellan stacken och heapen. Vad lagras på respektive ställe?
(2p): Stacken lagrar de lokala variablerna och returadresser i samband med funktionsanrop. Alla variabler på stacken har fix storlek. Minnet ”städas” automatiskt.
Heapen lagrar dynamisk data, programmeraren sköter minneshanteringen.

d) Beskriv en fördel med dynamiskt allokerat minne.
(1p): 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.

e) Vad har funktionen `bind()` för uppgift när vi programmerar sockets?
(1p): Kopplar en IP-adress och ett portnummer till lyssnings-socketen som den kommer acceptera uppkopplingsförfrågningar på.

Uppgift 2[4p]

Vilka värden har följande uttryck i C (vad evakuerar varje uttryck till)?
För samtliga uttryck gäller: `int x=4, y=26, z=45, a=1, b=9;`

a) `z > 8 && x != y / 2`
(1p): `1 && 4 != 13  1 && 1  1`

b) `!(a == x || a == y) && (b > x && a)`
(1p): `!(0 || 0) && (1 && 1)  !0 && 1  1`

c) `z >= b * 5 && y == x * 6 + 2 && !a`
(1p): `45 >= 45 && 26 == 26 && 0  1 && 1 && 0  0`

d) `x + a > b || z - y < b + x && z > y`
(1p): `0 || 19 < 13 && 1  0 || 0 && 1  0 || 0  0`

Uppgift 3 [3p]

Antag följande rad kod: `char str[50] = "Jag lär mig programmera C";`

a) Vilka index (från – till) har arrayen?
(0.5p): 0-49

b) Vad finns på index 25 i arrayen?
(0.5p): \0

c) Vilken längd har strängen, dvs. vad returnerar `strlen(str)`?
(0.5p): 25

d) Vilken storlek har arrayen, dvs. vad returnerar `sizeof(str)`?
(0.5p): 50

e) Hur många bitar (bits) allokeras på stacken för str?
(1p): 400 bits (50 byte * 8 bits/byte)

Uppgift 4 [8p]

Skapa en egendefinierad typ (struct) för ett bankkonto. Den nya typen ska innehålla följande information

- Clearingnummer (4 siffor)
- Kontonummer (9 siffror)
- Kontoinnehavare (för- och efternamn)
- Saldo (ska kunna innehålla ören)

a) Skriv kod som definiera den nya typen (structen).

b) Skriv kod för att skapa ett fält med 10 poster (array) av den nya typen genom att använda funktionen `malloc()`. Felhantering för allokering av dynamiskt minne är inget krav.

c) Skriv en kodrad som tilldelar värdet 398.50 till fältet ”saldo” i den tredje posten. Använd punktoperatorn.

d) Skriv en kodrad som tilldelar kontonumret 959500123 till fältet ”kontonummer” i sista posten. Använd piloperatorn.

Lösningar

a) (2p):
```
typedef struct{
   int clearingnummer;
   int kontonummer;
   char namn[25];
   float saldo;
}bankkonto;
```

b) (2p):
```
bankkonto *konton= malloc(sizeof(bankkonto)*10);
//bankkonto *konton= calloc(10,sizeof(bankkonto));
```

c) (2p):
```
konton[2].saldo = 398.50;
```

d) (2p):
```
(konton + 9)->saldo = 959500123;
```

alt a) (2p):
```
struct bankkonto{
   int clearingnummer;
   int kontonummer;
   char namn[25];
   float saldo;
};
```

alt b) (2p):
```
struct bankkonto *konton= malloc(sizeof(bankkonto)*10);
//struct bankkonto *konton= calloc(10,sizeof(bankkonto));
```

Uppgift 5 [4p]

Skriv om alla if-else-if-satser i koden nedan till en switch-case-konstruktion.

```
int main(void)
{
    int val = 0;

    puts("          MENY");
    puts("-------------------------------");
    puts("1 - Lägg till post");
    puts("2 - Ta bort post");
    puts("3 - Sök efter post");
    puts("4 - Skriv ut samtliga poster");
    puts("5 - Avsluta programmet");


    scanf("%d", &val);

    if(val == 1){
        laggTill();
    }
    else if(val == 2){
        taBort();
    }
    else if(val == 3){
        sok();
    }
    else if(val == 4){
        skrivUt();
    }
    else if(val == 5){
        avsluta();
    }
    else {
        puts("Felaktigt val");
    }

    return 0;
}
```

Skriv om denna del

Totalt (4p)

```
switch(val){                                //(1p) för rätt switch
    case 1: laggTill(); break;              //(1p) för rätt case
    case 2: taBort(); break;
    case 3: sok(); break;                   //(1p) för rätt användning av break
    case (4): skrivUt(); break;
    case (5): avsluta(); break;
    default: puts("Felaktigt val");         //(1p) för rätt default
}
// övriga fel (-0,5p per fel)
```

Uppgift 6 [10p]

Delfråga a till b nedan handlar om följande programkod.

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

void swap(int *x, int *y);
void printArray(int arr[]);

int main(void){
    int arr[STORLEK] = {}
    int i, j;

    printf("Mata in %d tal\n", STORLEK);
    for(i = 0; i <= STORLEK; i++)
        scanf("%d", arr[i]);

    printf("Arrayen före sortering:");
    printArray(arr);

    for(; i < STORLEK-1; i++)
        for(j = STORLEK-1; j>0; j--)
            if(arr[j] < arr[j-1])
                swap(arr[j-1], arr[j]);

    printf("Arrayen efter sortering:");
    printArray(arr);

    return 0;
}

void swap(int *x, int *y){
    int temp = *x;
    x = y;
    *y = temp;
}

void printArray(int arr[]){
    int i;
    for(i = 0; i < STORLEK; i++)
        printf("%d  ", arr[i]);
}
```

a) I koden finns ett antal fel (tankefel och fel som av en kompilator ger error). Hitta felen, skriv ner hela den felaktiga raden och visa hur den ska rättas till (ändras).
Påståenden om fel som i själva verket är korrekt kod ger poängavdrag (lägsta poäng är 0).

(0.5p): `int arr[STORLEK] = {}` `//semikolon (;) saknas`
(0.5p): `for(i = 0; i <= STORLEK; i++)` `//ska vara <`
(0.5p): `scanf("%d", arr[i]);` `//måste vara &arr[i]`
(0.5p): `for(; i < STORLEK-1; i++)` `//i=0 saknas`
(0.5p): `swap(arr[j-1], arr[j]);` `//& saknas på argumenten`
(0.5p): `x = y;` `//pekare, ska vara *x = *y`

b) Beskriv vad programmet utför. dvs. vad matas in och vad blir resultatet (inte rad för rad).
(2p): Användaren matar in 10 tal som sedan sorteras och skrivs ut i ordningen lägsta till högsta tal. En bubbelsortering.

c) Ge ett exempel från koden på en variabeldeklaration om det finns.
(1p): `int i, j;` Dvs. någon av variabeldeklarationerna från koden. Vi pratar inte om funktionsdeklarationer som är något annat.

d) Ge ett exempel från koden på en variabelinitiering om det finns
(1p): `int arr[STORLEK] = {};` eller `int temp = *x;` eller `i=0;`

e) Ge ett exempel från koden på en definition om det finns.
(1p): Finns ingen i koden eller är #define en definition? (dock kallar man även implementationen av funktionen för funktionsdefinition i bland).

f) Vad är skillnaden mellan att lägger en implementation av en funktion före eller efter funktionen `main()`?
(2p): Om man vill lägga en funktion efter main() kräver kopilatorn att man lägger till en funktionsdeklaration (funktionsprototyp) före main(). Läggs hela koden för funktionen (även kallat funktionsdefinitionen) efter main() behövs ingen funktionsdeklaration.

(En funktionsdeklaration berättar för kompilatorn antalet argument och av vilken typ argumenten är. Den berättar dessutom vilken typ av värde (om något) som funktionen returnerar.)

Uppgift 7 [4p]

Skriv kod för två olika loop-konstruktioner (två olika main-funktioner) som implementerar följande beskrivning:

För årets alla månader under ett år, summera alla månadernas motsvarande tal. I loopen ska alla tal som summeras skrivas ut med ett + mellan talen. Avsluta med ett = och summan. (dvs. 1+2+3+ … +12=78).

a) Använd en for-loop (2p):

```
int main(void){                 // Om main saknas -0,5p
    int i, summa=0;
                                // Om mer än en loop -0,5p
    for(i=1; i<=12; i++){
        if(i==12)
            printf("%d= ",i);    //Utskrifter av talen utanaför
        else                     //loopen -2p
            printf("%d+",i);
        summa+=i;
    }
    printf("%d",summa);     // …12+ =78 (plus och lika med)-0,5p
}
```

b) Använd en do-while-loop (2p):

```
int main(void){
    int i=1, summa=0;

    do{
        if(i==12)
            printf("%d= ",i);   //Utskrifter av talen utanaför
        else                    //loopen -2p
            printf("%d+",i);

        summa+=i;
        i++;
    }while(i<=12);
    printf("%d",summa);     // …12+ =78 (plus och lika med)-0,5p
```

```
}
```

Uppgift 8 [5p]

Skriv ett program som bara innehåller `main()` och en funktion `create_number()`.

`create_number()`:Funktionens uppgift är att allokera minnesplats på heapen motsvarande typen `long` och spara ett slumptal mellan 1 och 15000 på den platsen. Funktionen skall inte ta några inparametrar.

`main()`: Här ska funktionen `create_number()` anropas en gång, talet från funktionen ska dubbleras och skrivas ut på skärmen.

Felhantering för allokering av dynamiskt minne är ett krav i denna uppgift.

(5p):

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

long *create_number(void){ //(0,5p)
    long *number;

    srand(time(NULL)); //(0,5p)

    number = malloc(sizeof(long)); //(0,5p)
    if(number == NULL){
        printf("Error when trying to allocate memory!"); //(0,5p)
        exit(-1);
    }
    *number = rand()%15000+1; //(0,5p)

    return number; //(0,5p)
}

int main(){
    long *number=NULL;

    number = create_number(); //(0,5p)
    *number = *number * 2; //(0,5p)
    printf("%ld",*number); //(0,5p)

    free(number); //(0,5p)

    return 0;
}
```