Programmering, grundkurs, 8.0 hp HI1024, HI1900 etc., Tentamen TEN1

Måndagen den 10 januari 2011, 8.15 – 12.15

Tentamen består av två delar, del A och del B. Del A innehåller 10 kryssfrågor på olika teman inom *C*-programmering. Varje fråga är uppbyggd på ett påstående. Påståendet är antingen *helt korrekt* och är då alltså SANT, eller så finns *minst ett allvarligt fel* i påståendet och då är alltså påståendet FALSKT. Man kryssar det man tror på och om man kryssar rätt får man 1 poäng. Kryssar man fel får man –1 poäng. Det är alltså riskabelt att chansa. Man har alltid alternativet att kryssa AVSTÅR som då ger 0 poäng. Del A kan inte ge mindre än 0 poäng totalt. (Om man har mer fel är rätt får man alltså inga negativa poäng.) Del B innehåller 13 traditionella frågor som ni ska svara skriftligt på. Totalt antal poäng är 26. Gränsen för E är 13 och gränsen för Fx är 11. KTHs allämnna regler för examination gäller.

Del A.

Påstående 1. Då ett *C*-program kompileras så översätts källkoden till maskinkod som är körbar för en speciell datortyp. Olika fel kan då uppstå, *lexikala*, *syntaktiska* och *logiska*. Lexikala och syntaktiska fel kan inte rättas av kompilatorn, men vissa typer av logiska fel kan rättas. Programmeraren behöver då inte själv rätta dessa fel, det sker automatiskt.

[] SANT [] FALSKT [] AVSTÅR

Påstående 2. Ett *C*-program som innehåller kompileringsfel kan köras men det går mycket långsammare.

[] SANT [] FALSKT [] AVSTÅR

Påstående 3. Pekare i *C* är variabler som innehåller adresser till andra variabler, det går alltså teoretiskt att lagra ett pekares värde i en vanlig heltalsvariabel (`int` eller `long int`).

[] SANT [] FALSKT [] AVSTÅR

Påstående 4. Om en array har 10 platser numreras dessa från 0 till 9. Om man försöker skriva till plats nr 10 (som ju inte finns) så börjar det om från plats 0 igen, det vill säga det värde man försöker skriva in på plats 10 hamnar på plats 0 istället.

[] SANT [] FALSKT [] AVSTÅR

Påstående 5. Antag att vi har deklarationerna

```
struct person {
  char namn[20];
  int alder;
};

struct person p1, p2;
```

Om vi då lägger in data i `p1` och skriver `p2 = p1;` så kopieras data i `p1` in på de rätta platserna i `p1`. Vi behöver då alltså inte använda `strcpy()` för kopiering av strängfältet `namn` från `p1` till `p2`.

[] SANT [] FALSKT [] AVSTÅR

Påstående 6. Strängar, så som de hanteras i biblioteket `string.h`, är representerade som en array av characters (`char`) som avslutas med tecknet med koden 0. Man kan förstås bilda arrayer av characters som *inte* avslutas med tecknet med koden 0, men dessa arrayer kommer då inte att kunna hanteras som strängar av biblioteket `string.h`. Man kan till och med hävda att de noga taget inte kan kallas "strängar", i strikt mening, utan att de då bara är arrayer av characters.

[] SANT [] FALSKT [] AVSTÅR

Påstående 7. En funktion som anropas i *C* kan aldrig ändra på värdet av sina parametrar så att det märks utanför funktionen. Däremot kan man skicka adresser till saker som ligger utanför funktionen som funktionen kan ändra på, det kan då märkas utanför funktionen.

[] SANT [] FALSKT [] AVSTÅR

Påstående 8. Förprocessordirektiv kan hjälpa oss att definera värden i ett *C*-program som ska gälla överallt i programmet. Vi kan skriva

```
#define MAX 10
```

så blir `MAX` lika med 10 i hela programmet. Vi kan också, i den efterföljande programkoden, skriva `MAX = 20;` så tilldelas `MAX` värdet 20 istället för 10, ungefär som en variabel.

[] SANT [] FALSKT [] AVSTÅR

Påstående 9. Heltalsvariabler av typen `int` är normalt på 4 bytes (på en 32-bitarsdator), det betyder att de kan lagra heltal mellan ca –2 miljarder upp till +2 miljarder. Om man däremot deklarerar en variabel av typen `unsigned int` kan den lagra heltalsvärden i intervallet 0 till ca +4 miljarder.

[] SANT [] FALSKT [] AVSTÅR

Påstående 10. Om man begär att en division med noll ska utföras avbryts normalt programkörningen och ett fel genereras, "division by zero". Man kan dock undvika att programmet avbryts vid en division med noll om man ser till att den nolla man ska dividera med är inlagd i en flyttalsvariabel (`float` eller `double`), flyttalsvariabler har *variabel* precision vilket innebär att de aldrig riktigt har rätt värden. En division med noll blir då en division med något som nästan är noll och det kan gå igenom utan att programmet avbryts. Följande programkod kan alltså gå att köra:

```
float x = 0.0;
float y = 1.0/x;
```

[] SANT [] FALSKT [] AVSTÅR

Del B.

Uppgift 11. (1p) Studera nedanstående program:

```
#include <stdio.h>
main()
{
  int i,j;
  for(i=0;i<5;i++)
  {
    for(j=0;j<i;j++)
      printf("(%d,%d) ",i,j);
    printf("\n");
  }
}
```

Ange den utskrift som uppkommer då man kör det.

Uppgift 12. (1p) Studera nedanstående program:

```
#include <stdio.h>

main()
{
  char str[] = "ORVBA";
  int i, pos = 0;

  for(i=0;i<5;i++)
  {
    pos+=3; pos%=5;
    printf("%c", str[pos]);
  }
  printf("\n");
}
```

Ange den utskrift som uppkommer då man kör det.

Uppgift 13. (1p) Studera nedanstående program:

```
#include <stdio.h>

main()
{
  int i, a[] = {3, 2, 4, 1, 0};
  int b[] = {3, 2, 4, 1, 0};

  for(i=0;i<=4;i++)
    printf("%d ", a[b[i]] );
  printf("\n");
}
```

Ange den utskrift som uppkommer då man kör det.

Uppgift 14. (1p) Studera nedanstående program:

```
#include <stdio.h>

int f1(int a)
{
  a=10-a;
  printf("F1 %d\n",a);
  return a;
}

f2(int x, int *py)
{
  int z;
  *py++; x++;
  printf("F2 %d %d\n", x, *py);
  z=f1(x+*py);
}

main()
{
  int i=10,j=20;
  f2(i,&j);
  printf("MAIN %d %d\n", i, j);
}
```

Ange den utskrift som uppkommer då man kör det.

Uppgift 15. (2p) Studera nedanstående program, det är tänkt att användas för att omvandla temperaturangivelser angivna i Celsius och Fahrenheit enligt formlerna f = 9/5*c+32 respektive c=5/9*(c-32).

```
#include <stdio.h>

main()
{
  float c, f;
  int val = 0;

  while(val!=1||val!=2)
  {
    printf("Konvertera \n");
    printf("1) c->f eller 2) f->c: ");
    scanf("%d",val);
  }

  if(val==1)
  {
    printf("c->f\n");
    printf("Ange c: ");
    scanf("%f", &f);
    printf("f: %f\n", 9/5*c+32);
  }
  else
```

```
  {
    printf("f->c\n");
    printf("Ange f: ");
    scanf("%f", &f);
    printf("c: %f\n", 5/9*(f-32));
  }
}
```

Programmet har fem felaktiga rader. Rätta dessa fel. Felen är av logisk karaktär så det genereras inga kompileringsfel, dock genereras en varning.

Uppgift 16. (3p) Betrakta nedanstående program, det läser in 10 flyttal från en textfil.

```
#include <stdio.h>

main()
{
  float flyttal[10];
  int i=0,j;
  FILE * fp;

  fp = fopen ("flyttal.txt","r");

  while(i<10)fscanf(fp,"%f",&flyttal[i++]);

  /*...*/

  for(i=0;i<10;i++)printf("%f ", flyttal[i]);

}
```

I positionen som är markerad med /*...*/, lägg till en sortering som är uppbyggd på följande sätt:

1. Läs in alla tal i en vektor (redan gjort i `flyttal[10]`).
2. Sortera genom att gå igenom hela arrayen och hitta största och minsta talet, placera det minsta först i arrayen och det största sist.
3. Utför steg 2 igen fast på den del av arrayen som blir kvar om man bortser från första och sista elementet.
4. Fortsätt så tills hela arrayen är sorterad.

Uppgift 17. (1p) Man önskar utskriften

```
1 2 4 8 16
```

av programsnutten

```
i=...;
while(i<...)
{
  printf("%d ", i);
  i=...;
}
```

Fyll i *C*-kod där det står punkter så att programsnutten ger det resultat som är önskat.

Uppgift 18. (2p) Skriv ett program som läser in enskilda tecken i alfabetet, `a-z`, från tangentbordet. Varannan bokstav som matas in ska byta *case*, det vill säga om det är en stor bokstav så ska den bli liten och om det är en liten bokstav ska den bli stor. Resultatet ska skrivas ut på skärmen samtidigt med inmatningen. Ett körexempel:

```
> a
A
> B
B
> 8
> C
c
> d
d
```

De fetmarkerade tecknena är de som användaren matar in. I steg tre ser vi att användaren matar in en 8:a, det är inte en bokstav, då kommer prompten (>) tillbaka igen tills användaren matar in ett tecken, då matar användaren in `C` som omvandlas till `c`.

Uppgift 19. (2p.) I *Del A* förekom flera falska påståenden. Välj två av dessa falska påståenden och förklara vad som egentligen gäller i sammanhanget, det vill säga uttryck vad som skulle vara sant i sammanhanget och varför det är på det viset.

Uppgift 20. (2p) Strängvändaren. Nedanstående program är tänkt att ta en sträng där ett ord lagras som användaren matar in. Programmet fungerar inte som det är tänkt. Förklara vad problemet är och rätta felet genom att skriva om väl valda rader.

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

main()
{
  char str[100];
  int i,len;

  printf("Str: "); gets(str); printf("\n");
  len = strlen(str);
  printf("Str: %s Len: %d\n\n", str, len);
  for (i=0;i<=len;i++)str[i]=str[len-i];
  printf("Str: %s Len: %d\n", str, len);
}
```

En provkörning:

```
Str: Johnny

Str: Johnny Len: 6

Str:  Len: 6
```

Vi vill ju att sista utskriften ska vara `Str: ynnhoJ  Len: 6`