ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR TEKNIK

Lösningarna till tentamensuppgifterna sätts ut på kurssidan på nätet i dag kl 19.

Denna tenta kommer att vara färdigrättad Må 22/8 och kan då hämtas på mitt tjänsterum T2221 mellan kl 13 och 15. Vid detta tillfälle har du också möjlighet att diskutera rättningen. Tentor som ej hämtats då placeras på studentexpedition och då har du ingen möjlighet att klaga på rättningen.

Tentamen i Objektorienterad Programmering 5p, Au, D, Fri, Pr, 050817

Hjälpmedel	: Inga		
Tid	: 14 - 19		
Ansvarig lärare	: Gunnar Joki	Tel :	arb 303317
			hem 274825
			mob 070-5474825

Svar till samtliga uppgifter 1-15 ska skrivas på utdelat extra papper. Använd ett papper till uppgifterna 1-5, två papper till uppgifterna 6-10 och ett papper per uppgift till uppgifterna 11-15 Skriv din tentamenskod på varje inlämnat extra papper.

Den maximala poängen för respektive uppgift står angiven efter uppgiftens nummer. Totalt kan 40 poäng erhållas. För betyget 3 krävs ca 20, för betyget 4 ca 28 och för betyget 5 ca 34 poäng.

Om inget speciellt anges gäller frågorna Microsoft C++.

Detta häfte ska du behålla.

Lycka till!

1) (1p) I specifikationen för klassen TRIANGEL finns de privata medlemsvariablerna bas och hojd definierade enligt:

```
float bas, hojd;
```

Implementera konstruktorn nedan så att den kopierar parametervärdena till motsvarande medlemsvariabler. Specifikation av konstruktorn enligt:

```
TRIANGEL( float bas = 0.0, float hojd = 0.0);
```

---

2) (1p) Klassen TRIANGEL har en medlemsfunktion, area med specifikationen,

```
float area(float bas, float hojd);
```

Implementera denna funktion.

---

3) (1p) Skriv de satser som skapar ett objekt av klassen TRIANGEL ovan, med basen 8.6 och höjden 2.3 och som skriver ut triangelns area.

---

4) (1p) Inmatningsoperatorns specifikation för TRIANGEL-klassen ser ut som:

```
friend istream &operator>>(istream &in, TRIANGEL &t);
```

Vad betyder &-tecknet som finns framför de båda parameternamnen?

---

5) (1p) Implementera inmatningsoperatorn enligt uppgift 4 ovan.

---

6) (2p) Klassen avbildar tentamina enligt:

```
// Specifikation av klassen TENTAMEN – Tentamen.h

class TENTAMEN
{
   private:
      int kod;          // Tentamenskoden
      float poang;      // Tentamensresultat i poäng
   public:
      TENTAMEN(int kod = 0, float poang = 0.0);
      void las();       // Läser tentamensdata med ledtexter
      void skriv();     // Skriver tentamensdata med ledtexter
      int get_kod();    // Returnerar koden
      float get_poang();// Returnerar poängen
      char get_betyg(float gr3, float gr4, float gr5);
      // Returnerar betyget U, 3, 4, 5 om gränsen för 3 är gr3 osv
};
```

Implementera medlemsfunktionen las.

7) (2p) Implementera medlemsfunktionen skriv i uppgift 6 ovan, så att den skriver ut tentamensdata med ledtexer.

---

8) (2p) Implementera medlemsfunktionen get_betyg i uppgift 6 ovan.

---

9) (2p) Skriv ett huvudprogram som skapar en vektor av 5 trianglar med baserna 1.0, 2.0, 3.0, 4.0, 5.0 och höjderna 4.5, 3.5, 2.5, 1.5, 0.5 av klassen TRIANGEL ovan och skriver ut trianglarnas areor.

---

10)(2p)Istället för att ha funktionen skriv kan man använda sig av en utskriftsoperator i klassen TENTAMEN ovan. Implementera en sådan.

---

11)(5p)Skriv ett huvudprogram som börjar med att fråga efter betygsgränserna på en tentamen och hur många tentamensresultat, som ska läsas in. Programmet läser sedan in tentamensresultatet med kod och poäng enligt uppgift 6 ovan och skriver ut fördelningen av betygen genom att skriva ut antalet erhållna U, 3, 4 och 5.

---

12)(5p)Klassen PERSON avbildar personer enligt:

```
// Specifikation av klassen PERSON – Person.h

class PERSON
{
   private:
      char persnr[12];
      char namn[30];
   public:
      PERSON(char *persnr = ””, char *namn = ””);
      void las();       // Läser personens data med ledtexter
      void skriv();     // Skriver personens data med ledtexter
};
```

En STUDENT är en person med studiepoäng enligt:

```
// Specifikation av klassen STUDENT -- student.h
#include ”Student.h”
class STUDENT : public PERSON
{
   private:
      float poang;                  // Studiepoäng ex 35.5
   public:
      STUDENT(char *persnr = ””, char *namn = ””, float poang = 0.0);
      void las();                   // Läser data med ledtexter
      void skriv();                 // Skriver data med ledtexter
      float get_poang();            // Returnerar studiepoängen
      void set_poang(float dp);     // Ökar studiepoängen med dp
};
```

Implementera alla medlemsfunktioner i klassen STUDENT om PERSON är given.

13)(5p)Klassen KURS nedan avbildar kurser som en student läser

```
// Specifikation av klassen KURS – Kurs.h

#include ”Student.h”
#include ”Tentamina.h”

class KURS
{
   private:
      char namn[30];          // Kursens namn
      STUDENT *sp;            // Student som läser kursen
      TENTAMEN *tp;           // Kursen avslutas med en tentamen
   public:
      KURS(char *namn, STUDENT *sp, TENTAMEN *tp);
      void las();            // Läser kursdata med ledtexter
      void skriv();          // Skriver kursdata med ledtexter
};
```

Implementera alla medlemsfunktioner för klassen KURS.

---

14)(5p)I textfilen Kurser.txt finns alla kurser inskrivna som en student är registrerad på. Först står studentens namn följt av antalet kurser och sedan kommer alla kurser med kursnamn betyg och poäng, radvis enligt:

```
. . . . . .

Kalle.Anka  8
OPdel1      3     3.0
ProgCdel1   4     4.0
Prmetdel1   3     3.5
Matte1      U     5.0
Matte2      X     5.0
OOPdel2     3     2.0
ProgCdel2   G     1.0
Prmetdel2   G     1.5

. . . . . .
```

Betyget kan vara X för saknad, U för underkänd, G för Godkänd eller 3, 4 , 5. Skriv ett huvudprogram som läser in en students namn och som letar upp namnet i filen och skriver ut studentens totala studiepoäng alltså summan av alla kurspoäng utom dom som har betyget X eller U.

15)(5p)Klassen KVADKONST avbildar kvadratkonst som byggs upp av ett antal lika stora kvadrater. Implementera alla medlemsfunktioner i klassen KVADKONST om klassen KVADRAT är given enligt:

```
// Specifikation av klassen KVADRAT -- Kvadrat.h

class KVADRAT
{
   private:
      float sida;

  public:
      KVADRAT(float sida);
};


// Specifikation av klassen KVADKONST – Kvadkonst.h

#include ”Kvadrat.h”

class KVADKONST
{
   private:
      KVADRAT *kvp;     // Pekare till första kvadraten
      int nr;           // Antalet kvadrater
   public:
      KVADKONST(int nr = 0);
      ¨KVADKONST();
      KVADKONST(const KVADKONST &kk);
      const KVADKONST &operator=(const KVADKONST &kk);
};
```

Lösningar till tentamen i Objektorienterad Programmering 5p, 050817

1)
```
TRIANGEL::TRIANGEL( float bas = 0.0, float hojd = 0.0)
{
   this->bas = bas;
   this->hojd = hojd;
}
```

2)
```
float TRIANGEL::area()
{
   return bas*hojd/2;
}
```

3)
```
TRIANGEL t(8.6, 2.3);

cout << t.area() << endl;
```

4) Markerar att de formella parametrarna är referensparameter vilket innebär att de är namn på samma minnesutrymme som de aktuella parametrarna.

5)
```
istream &operator>>(istream &in, TRIANGEL &t)
{
   cout << ”Ge triangelns bas : ”;
   in >> t.bas;
   cout << ”Ge triangelns höjd : ”;
   in >> t.hojd;
   return in;
}
```

6)
```
void TENTAMEN::las()
{
   cout << "Ge kod : ";
   cin >> kod;
   cout << "Ge poäng : ";
   cin >> poang;
}
```

7)
```
void TENTAMEN::skriv()
{
   cout << "Tentamenskod : ” << kod << endl;
   cout << "Tentamenspoäng : ” << poang << endl;
}
```

8)
```
char TENTAMEN:: get_betyg(float gr3, float gr4, float gr5)
{
   if (poang < gr3)
      return ’U’;
   if (poang < gr4)
      return ’3’;
   if (poang < gr5)
      return ’4’;
   return ’5’;
}
```

9)
```
void main()
{
   TRIANGEL tv[5] = { TRIANGEL(1.0, 4.5), TRIANGEL(2.0, 3.5),

                      TRIANGEL3.0, 2.5), TRIANGEL(4.0, 1.5),
                      TRIANGEL(5.0, 0.5)};

   for (int i = 0; i < 5; i++)
   {
       cout << ”Area för triangel ” << i+1 << ” : ” << tv[i].area() ;
   }
}
```

10)
```
ostream &operator<<(ostream &ut, TENTAMEN t)
{
   ut << ”Kod : ”;
   ut << t.kod;
   ut << ”Poäng : ”;
   ut << t.poang;
   return ut;
}
```

11)
```
#include <iostream>
#include <conio.h>
#include ”Tentamen.h”

using namespace std;

void main()
{
   float gr3, gr4, gr5;
   int nr, nr_U = 0, nr_3 = 0, nr_4 = 0, nr_5 = 0;
   TENTAMEN t;

   cout  << ”Ge gränser för 3, 4 och 5 : ”;
   cin >> gr3 >> gr4 >> gr5;

   cout << ”Hur  många tentamina ska inmatas : ”;
   cin >>  nr;

   for(int i = 1; i <= nr; i++)
   {
      t.las();
      switch  (t.get_betyg(gr3, gr4, gr5))
      {
         case ’U’ : nr_U++;break;
         case ’3’ : nr_3++;break;
         case ’4’ : nr_4++;break;
         case ’5’ : nr_5++;break;
      }
   }
   cout << ”Antal U : ” << nr_U << endl;
   cout << ”Antal 3 : ” << nr_3 << endl;
   cout << ”Antal 4 : ” << nr_4 << endl;
   cout << ”Antal 5 : ” << nr_5 << endl;

   getch();
}
```

12)
```
STUDENT::STUDENT(char *persnr, char *namn, float poang)
                  : PERSON(persnr, namn)
{
   this->poang = poang;
}

void STUDENT::las()
{
   PERSON::las();
   cout << ”Ge poäng : ”);
   cin >> poang;
}

void STUDENTskriv()
{
   PERSON::skriv();
   cout << ”Poäng : ” << poang << endl;
}

float STUDENT::get_poang()
{
   return poang;
}

void STUDENT::set_poang(float dp)
{
   poang += dp;
}
```

13)
```
#include ”Student.h”
#include ”Tentamen.h”

KURS::KURS(char *namn, STUDENT *sp, TENTAMEN *tp)
{
   strcpy(this->namn, namn);
   this->sp = sp;
   this->tp = tp;
}

void KURS::las()
{
   cout << ”Ge kursnamn : ”;
   cin >> namn;
   sp->las();
   tp->las();
}

void KURS::skriv()
{
   cout << ”Kursnamn : ” << namn << endl;
   sp->skriv();
   tp->skriv();
}
```

14)
```
#include <conio.h>
#include "Kurs.h"
#include <fstream>
#include <cstring>
using namespace std;
```

```
void main()
{
   ifstream tsin(”Kurser.txt”);
   float sum_poang = 0.0;
   int nr;
   char soknamn[30], studnamn[30];
   char kursnamn[20];
   char betyg;
   cout << ”Vem söks?”;
   cin.getline(soknamn, sizeof(soknamn));
   while ((tsin >> studnamn >> nr)&&(stricmp(soknamn, studnamn)!=0)
   {
   }
   for( i = 1; i <= nr; i--)
   {
      tsin >> kursnamn >> betyg >> poang;
                                {
      if (betyg == ’G’ || betyg == ’3’ ||
          betyg == ’4’  || betyg == ’5’)

      sumpoang += poang;
   }
   cout << ”Total studiepoäng : ” << sum_poang;
   getch();
}
```

15)

```
KVADKONST(int nr = 0)
{
   this->nr = nr;
   this->kvp = new KVADRAT[nr];
}

KVADKONST::¨KVADKONST()
{
   delete [] kvp;
   nr = 0;
}

KVADKONST::KVADKONST(const KVADKONST &kr)
{
   kvp = new KVADRAT[kr.nr];
   for (int i = 0; i < kr.nr; i++)
   {
      kvp[i] = kr.kvp[i];
   }
   nr = kr.nr;
 }

const KVADKONST &operator=(const KVADKONST &kr)
{
   if ( this != &kr )
   {
      delete [] kvp;
      kvp = new KVADRAT[kr.nr];

      for (int i = 0; i < kr.nr; i++)
      {
         kvp[i] = kr.kvp[i];
      }
      nr = kr.nr;
   }
   return *this;
}
```