ÖREBRO UNIVERSITET

INSTITUTIONEN FÖR TEKNIK

Lösningarna till tentamensuppgifterna sätts ut på kurssidan på nätet i dag kl 13.

Denna tenta kommer att vara färdigrättad Fr 28/1 och kan då hämtas på mitt tjänsterum T2221 mellan kl 15 och 17. 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.

Omtentamen i Objektorienterad Programmering 5p, Au, D, E, Fri, Pr, 050122

Hjälpmedel	: Inga		
Tid	: 08 - 13		
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 Borland C++.

Detta häfte ska du behålla.

Lycka till!

1) (1p) I specifikationen för klassen GPS finns de privata medlemsvariablerna x, y, z definierade enligt:

```
float x, y, z;    // Koordinaterna x, y i planet och z i höjdled.
```

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

```
GPS( float x = 0.0, float y = 0.0, float z = 0.0);
```

---

2) (1p) Skriv de satser som skapar ett objekt av klassen GPS ovan, med x = 876.6, y = 345.7 och z = 12 .3.

---

3) (1p) Antag att klassen GPS har en egen utmatningsoperator och inmatningsoperator definierade. Skriv ett huvudprogram som skapar två GPS-objekt och skriver ut dessa i omvänd ordning jämfört med inmatningen.

---

4) (1p) Utmatningsoperatorns specifikation ser ut som:

```
friend ostream &operator<<(ostream &ut, GPS g);
```

Varför markerar man operatorn som friend ?

---

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

---

6) (2p) Klassen FASTIGHET avbildar fastigheter enligt:

```
// Specifikation av klassen FASTIGHET – Fastighet.h

class FASTIGHET
{
   private:
      char bet[20];     // Fastighetens beteckning ex. Ladan5
      int tax;          // Fastighetens taxeringsvärde i hela kr
      int pris          // Fastighetens försäljningspris i hela kr
   public:
      FASTIGHET(char *bet = ””, int tax = 0, int pris = 0);
      void las();       // Läser fastighetens data med ledtexter
      void skriv();     // Skriver fastighetens data med ledtexter
      int get_tax();    // Returnerar taxeringsvärdet för fastigheten
      int get_pris();   // Returnerar fastighetens försäljningspris
      float get_tax_pris_proc(); // Returnerar tax/pris i procent
};
```

Implementera medlemsfunktionen las.

---

7) (2p) Implementera medlemsfunktionen skriv i klassen FASTIGHET ovan, så att den skriver ut fastighetens data.

8 (2p) Implementera medlemsfunktionen

```
float get_tax_pris_proc(); // Returnerar tax/pris i procent.
```

---

9) (2p) Skriv ett huvudprogram som skapar och läser in data till två fastighetsobjekt av klassen FASTIGHET ovan och skriver ut dem i ordning efter förhållandet mellan taxeringsvärdet och försäljningspriset, med den som har det lägsta förhållandet först.

---

10)(2p)Istället för att ha en las-funktion i klassen FASTIGHET kan man ha en inmatningsoperator. Implementera inmatningsoperatorn.

---

11)(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 mäklare är en person som säljer fastigheter med provision

```
// Specifikation av klassen MAKLARE – Maklare.h

#include ”Person.h”

class MAKLARE : public PERSON
{
   private:
      float prov;             // Provision ex.3.5% av priset
   public:
      MAKLARE(char *namn = ””, char *persnr= ””, float prov = 0.0);
      void las();            // Läser data med ledtexter
      void skriv();          // Skriver data med ledtexter
      float get_prov();   // Returnerar mäklarens provision i procent
};
```

Implementera alla medlemsfunktioner i klassen MAKLARE.

12)(5p)Klassen FASTAFF avbildar fastighetsaffärer förmedlade av en mäklare.

```
// Specifikation av klassen FASTAFF - Fastaff.h

#include "Fastighet.h"

class FASTAFF
{
   private:
      FASTIGHET *fp;          // Förmedlad fastighet
      MAKLARE *mp;            // Mäklare
   public:
      FASTAFF(FASTIGHET *fp=NULL, MAKLARE *mp=NULL);
      void las();             // Läser data med ledtexter
      void skriv();           // Skriver data med ledtexter
      int get_prov_kr();    // Returnerar provision i kr
};
```

Implementera alla medlemsfunktioner för klassen FASTAFF.

---

13)(5p)En mäklare har alla fjolårets fastighetsaffärer i en textfil, Fast2004.txt, med fastighetsbeteckning, taxeringsvärde och försäljningspris radvis enligt:

```
Ladan5      1450000     1780000
Loftet3     1560000     1980000
. . . . . . . . . . . . . . . .
```

Skriv ett huvudprogram som läser hela filen Fast2003.txt och skriver ut mäklarens totala provision under detta år om den utgör 2.5 % av försäljningspriset

---

14)(5p) En mobiltelefon kan avbildas med data och operationer enligt nedanstående specifikation. Implementera alla medlemsfunktioner i klassen MOBIL

```
//Specifikation av klassen MOBIL - Mobil.h

class MOBIL
{
   private:
      char *telnr;                  // Telefonnummer
      float total_samtalstid;       // Total samtalstid i minuter
      float pris_min;               // Samtalspris per minut
   public:
      MOBIL(char *telnr, float pris_min, float total_samtalstid);
      ~MOBIL();
      char *get_telnr();
      void set_total_samtalstid(int sekunder);
      float get_total_samtalstid();
      float samtals_kostnad(int sekunder);
};
```

OBS! Telefonnumret ska allokeras dynamiskt till samma som kommer in som parameter.

15)(5p)Ett mobilsamtal kan avbildas som ett objekt av klassen MOBSAM enligt:

```
class MOBSAM
{
   private:
      MOBIL *mp1, *mp2;
   public:
      MOBSAM(MOBIL *mp1, MOBIL *mp2);
};
```

Implementera konstruktorn och skriv ett huvudprogram som skapar ett mobilsamtal där den första mobilen ringer upp den andra. Samtalet kommer alltså att debiteras den första mobilen. Programmet ska med hjälp av funktionen clock bestämma samtalstiden och skriva ut vad samtalet kostar. Anropa funktionen sleep(5000) för att simulera samtalet som tar 5 sekunder. Glöm inte att uppdatera mobilens totala samtalstid. Låt ditt huvudprogram skriva ut den totala samtalstiden.

Hjälp!

Funktionen clock returnerar antalet sekunder som gått sedan 1970. Man kan avläsa klockan då samtalet startar och slutar. Skillnaden mellan dessa tider är samtalstiden som fås i sekunder om man dividerar tiden med **CLOCKS_PER_SEC** , som är en systemberoende konstant.

Lösningar till tentamen i Objektorienterad Programmering 5p, 050122

1)
```
GPS::GPS(int x, int y, int z)
{
   this->x = x;
   this->y = y;
   this->z = Z;
}
```

2)
```
GPS g(876.6, 345.7, 12.3);
```

3)
```
void main()
{
   GPS g1, g2;

   cin >> g1 >> g2;
   cout << g2 << endl << g1 << endl;
}
```

4) För att få direkt tillgång till privata medlemsdata för GPS-klassen.

5)
```
ostream &operator<<(ostream &ut, GPS g)
{
   ut << ”x = ” << g.x << endl;
   ut << ”y = ” << g.y << endl;
   ut << ”z = ” << g.z << endl;
   return ut;
}
```

6)
```
void FASTIGHET::las()
{
   cout << "Ge fastighetsbeteckning : ";
   cin.getline(bet, sizeof(bet));
   cout << "Ge taxeringsvärde : ";
   cin >> tax;
   cout << ”Ge försäljningspris : ”;
   cin >> pris;
   cin.ignore();
}
```

7)
```
void FASTIGHET::skriv()
{
   cout << "Fastighetens beteckning : ” << bet << endl;
   cout << "Taxeringsvärde : ” << tax << endl;
   cout << "Försäljningsvärde : ” << pris << endl;
}
```

8)
```
float FASTIGHET::get_tax_pris_proc()
{
   return (float)tax / pris * 100;
}
```

9)
```
void main()
{
   FASTIGHET f1, f2;

   f1.las();
   f2.las();
   if (f1.get_tax_pris() < f2.get_tax_pris())
   {
```

```
      f1.skriv();
      f2.skriv();
   }
   else
   {
      f2.skriv();
      f1.skriv();
   }
}
```

10)
```
istream &FASTIGHET::operator>>(istream &in, FASTIGHET &f)
{
   cout << ”Ge fastighetens beteckning : ”;
   in >> f.bet;
   cout << ”Ge taxeringsvärde : ”;
   in >> f.tax;
   cout << ”Ge försäljningspris : ”;
   in >> f.pris;
   return in;
}
```

11)
```
include ”Maklare.h”

MAKLARE::MAKLARE(char *persnr, char *namn, float prov)
                 :PERSON(persnr, namn)
{
   this->prov = prov;
}

void MAKLARE::skriv()
{
   PERSON::skriv();
   cout << ”Provision : ”;
   cout << prov << endl;
}

void MAKLARE::las()
{
   PERSON::las();
   cout << ”Ge provision : ”;
   cin >> prov;
}

void float get_prov()
{
   return prov;
}
```

12)
```
#include ”Fastighet.h”
#include ”Maklare.h”

FASTAFF::FASTAFF(FASTIGHET *fp, MAKLARE *mp)
{
   this->fp = fp;
   this->mp = mp;
}

void FASTAFF::las()
{
   fp->las();
   mp->las();
}
```

```
void FASTAFF::skriv()
{
   fp->skriv();
   mp->skriv();
}

int FASTAFF::get_prov_kr()
{
   return mp->get_prov() * fp->get_pris() / 100;
}
```

13)

```
#include <conio.h>
#include "Fastaff.h"
#include <iostream>
#include <fstream>
using namespace std;

void main()
{
   ifstream tsin("Fast2003.txt");
   char fastbet[20];
   int fasttax, fastpris, sum = 0;
   MAKLARE m("340804-8765", "Adam Anersson", 2.5);

   while (tsin >> fastbet >> fasttax >> fastpris)
   {
      FASTIGHET f(fastbet, fasttax, fastpris);
      FASTAFF a(&f, &m);

      sum += a.get_prov_kr();;
   }
   cout << "Total provision 2003 i kr : " << sum << endl;
   getch();
}
```

14)

```
#include "Mobil.h"
#include <cstring>

MOBIL::MOBIL(char *telnr, float pris_min, float total_samtalstid)
{
   this->telnr = new char[strlen(telnr)+1];
   strcpy(this->telnr, telnr);
   this->pris_min = pris_min;
   this->total_samtalstid = total_samtalstid;
      }

MOBIL::~MOBIL()
      {
         delete [] telnr;

}

char *MOBIL::get_telnr()
{
         return telnr;
}

void MOBIL::set_total_samtalstid(int sekunder)
{
   total_samtalstid += sekunder/60.0;
```

```
}

float MOBIL::get_total_samtalstid()
{
   return total_samtalstid;
}

float MOBIL::samtals_kostnad(int sekunder)
{
   return  sekunder / 60.0 * pris_min;
}
```

15)

```
#include <conio.h>
#include <ctime>
#include <Windows.h>

#include <iostream>
#include "Mobsam.h"

using namespace std;

void main()
{
   MOBIL m1("0701234567", 1.20, 34.5), m2("0702345678", 1.35, 45.6);
   MOBSAM ms (&m1, &m2);
   int start, slut, tid;

   start = clock();
   Sleep(5000);
   slut = clock();
   tid = (slut - start) / CLOCKS_PER_SEC;

   cout<<"Samtalet kostar "<<m1.samtals_kostnad(tid)<<" kr"<< endl;
   m1.set_total_samtalstid(tid);
   cout<<"Total samtalstid är "<<m1.get_total_samtalstid()<<" min";
   getch();
}
```