Statiskie klases locekļi

- Klases statiskie locekļi (atribūti) ir kopīgi visiem klases eksemplāriem (objektiem)
- Statiskā atribūta deklarācija nav tā definīcija – definīcija jāveic ārpus klases – klases realizācijas failā (parasti to apvieno ar inicializāciju)
- Statiskajiem klases atribūtiem var piekļūt, neizmantojot objektus

Statiskie klases locekļi (turpinājums)

DemoStat.h

```
class DemoStat
{public:
   static int n;         // tikai deklarācija !
   DemoStat() { n++; }
   ~DemoStat() { n--; }
};
```

DemoStat.cpp

```
#include "DemoStat.h"
int DemoStat::n = 0;  // definīcija un inicializācija
```

Main.cpp

```
void main()
{  DemoStat a, b[5];
   DemoStat* c = new DemoStat;
   cout << a.n << endl;           // izvada 7
   cout << b[4].n << endl;        // izvada 7
   delete c;
   cout << DemoStat::n << endl;// izvada 6, piekļūšana bez objekta
}
```

Statiskie klases locekļi (turpinājums)

- Klases metodes arī var būt statiskas

- Statiskajās klases metodēs nedrīkst lietot parastos klases locekļus un rādītāju `this`

Statiskie klases locekļi (turpinājums)

```
class Gamma
{  int m;
   static int ms;
public:
   static void comp(int, char);
};

int Gamma::ms = 0;

void Gamma::comp(int i, char c)
{
   //m = i;            // KĻŪDA!
   ms = i + c;         // OK
   //this->ms = i;     // KĻŪDA!
}

void main()
{
   int k = 3;
   Gamma::comp(k, 'A');   // OK, lai arī neviens objekts vēl nav radīts

   Gamma a;
   a.comp(k, 'B');        // OK
   //comp(13, 'C');       // KĻŪDA!
   Gamma::comp(12, 'D');  // OK
}
```

Statiskie klases locekļi (turpinājums)

- Statiskās funkcijas ērti lietot, lai radītu jaunus klases objektus – šādā funkcijā izsauc klases konstruktoru, radot objektu ar operāciju new
- Priekšrocības:
  - var pārbaudīt parametru pareizību pirms konstruktora izpildes
  - var atdot vērtību izsaucošajai funkcijai
  - var iztikt ar mazāku konstruktoru skaitu

```
class TriangC
{
   int a, b, c, color;
   static const double DEG2RAD;   // statiska konstante
   static const int max = 10;     // tikai veseliem tipiem
public:
   TriangC(int, int, int, int);
   static TriangC* make(int, int);
   static TriangC* make(int, int, double, int);
};
```

Statiskie klases locekļi (turpinājums)

```
#include "TriangC.h"
#include <math.h>
const double TriangC::DEG2RAD = 3.141592654 / 180;

TriangC::TriangC(int a, int b, int c, int color)
{ this->a = a; this->b = b; this->c = c; this->color = color;}

TriangC* TriangC::make(int side, int color)
{  if (side <= 0) return NULL;
   else return new TriangC(side, side, side, color);
}

TriangC* TriangC::make(int a, int b, double alpha_deg, int color)
{
   if (a <= 0 || b <= 0)
      return NULL;
   else
   {
      int c = (int)sqrt(a*a + b*b – 2*a*b*cos(DEG2RAD*alpha_deg));
      return new TriangC(a, b, c, color);
   }
}
```

Statiskie klases locekļi (turpinājums)

```
#include "TriangC.h"

void main()
{
   TriangC t0(3, 4, 5, 1);

   //t0 = TriangC::make(3, 1); // KĻŪDA !

   TriangC* t1;

   t1 = TriangC::make(3, 1);

   delete t1;

   t1 = TriangC::make(3, 4, 90, 1);

   delete t1;

   t1 = TriangC::make(-3, 1); // atgriež NULL

}
```

Klases draugi

- Klases draugs (`friend`) ir funkcija vai klase, kurai ir tiesības piekļūt klases `private` un `protected` locekļiem, kaut arī tā nav šīs klases loceklis

- Par klases draugu var deklarēt:
  - ārēju (t.i. globālu) funkciju
  - citas klases funkciju
  - citu klasi (visas šīs klases funkcijas)

Klases draugi (turpinājums)

- Draugs – citas klases funkcija

```
class X
{private:
   int g;
 public:
   void funX();
};
```

```
class Y
{private:
   int n;
   float p;
   friend void X::funX();
};
```

```
void X::funX()
{
   Y y;
   y.n = 2;
}
```

Lai arī klases draugus apraksta klases deklarācijā, tie tomēr nav klases locekļi.

Klases draugi (turpinājums)

- Draugs – globāla funkcija

```
class MyType
{private:
   int length;
 public:
   void setLength(int);
   friend void functEx(MyType*, int);
};

void MyType::setLength(int n)
{
   this->length = n;
}

void functEx(MyType* p, int k) // globāla funkcija, kas
{                              // nepieder nevienai klasei
   p->length = k;
}
```

Klases draugi (turpinājums)

- Draugs – cita klase

```
class Beta
{private:
   int b;
   float calcB();
 public:
   void clearB();
};
```

```
class Alpha
{private:
   int a;
   void functA();
   friend class Beta;
};
```

Funkcijās *calcB()* un *clearB()* var lietot klases *Alpha* locekļus *a* un *functA()*.

Piezīmes par draugiem

Klases draugus apraksta klases deklarācijā,
tomēr tie nav klases locekļi!

```
class MyType {
       private: int length;
       friend void funcEx(MyType*, int);
public:
       void setLength(int);
};

void MyType::setLength( int n ) // ir klases loceklis
{ length = n; }

void funcEx( MyType *p, int k )   // nav klases loceklis!
{ p->length = k; }
```

Šeit var lietot length tikai tāpēc, ka funkcija ir *friend* klasei MyType!

Klases draugi (turpinājums)

n Draudzība nav transitīva

n Draudzību nemanto