Bevezetés a programozásba 2

7. Előadás: Objektumszintű
és osztályszintű elemek, hibakezelés

ISMÉTLÉS

Osztály

```
class Particle {
public:
    Particle(int X, int Y);
    virtual void mozog( … );
    virtual void rajzol( … ) const;
protected:
    double x,y;
    unsigned char r,g,b;
};
```

ISMÉTLÉS

Fordítási egység

- Fordítási egység az, ami önállóan fordítható
- A C++ nyelvben minden .cpp fájl önálló fordítási egység
- Moduláris programozás
  - Felgyorsítja a fordítást: csak a megváltozott forráskódokhoz tartozó egységek fordulnak újra
  - A személyi felelősség értelmezhető a rendszer alkotóelemeire: önállóan tesztelhető modulok
  - Strukturális / moduláris programozás

A static kulcsszó

Sok jelentését a kontextus egyértelműsíti

- lokális változó deklarációja előtt
- globális változó deklarációja előtt
- tagfüggvény szignatúrája előtt
- mező deklarációja előtt

Általánosságban körülbelül a „maradandó, egyedüli” jelentést érdemes elképzelni

static lokális változó

Ha egy függvényben lokális változót deklarálunk, annak minden híváskor újra lefoglalódik a memória (a stack-en), kivéve, ha static

A static változó érvényessége szerint olyan, mint egy globális változó: nem szűnik meg a függvény végén, és értékét megtartja a program teljes futásán át

Azonban a static változó lokális láthatóság szempontjából.

static lokális változó

```
void fv() {
  static int count = 0;
  cout << ++count << endl;
}

int main() {
  for( int i = 0; i < 10; ++i ) {
    fv();
  }
}
```

static lokális változó

Lokális láthatóságú globális érvényességű változó

Érdekessége, hogy kivételesen kap kezdeti értéket: nullát

Kitérő: egy változó a memóriában három helyen lehet

- a stack-en: lokális változó
- a heap-en: dinamikusan lefoglalt változó
- a globális változók között (korlátos méretű, kinullázva inicializálódik)

static globális változó és függvény

ez az extern ellentéte, a fordítási egységből kifelé nem látszik a változó/függvény

Deprecated (nem javasolt) feature

név nélküli namespace van helyette C++-ban

```
static ofstream flog((string(__FILE__)
                                +".log").c_str());

void fv() {
   flog << "log: " << változó;
}
```

Osztályok és a static

Objektumszintű – osztályszintű

A static jelentése: osztályszintű

Ha mező: egyetlen példány van, közös az egész típusnak (vs minden példány saját mezővel rendelkezik)

Ha tagfüggvény: objektum nélkül hívható metódus

A statikus metódus objektum nélkül is hívható

```
class A {
public:
  A();
  static void sfv();
};

int main() {
  A::sfv();
  A a;
  a.sfv();
}
```

Statikus mező

```
class A {
public:
   A() {++count;}
   int getCount() const {return count;}
protected:
   static int count;
};

int A::count;

int main() {
   A a,b,c;
   cout << a.getCount();
}
```

Statikus mező

```
class A {
public:
   A() ;
   int getCount();
protected:
   static int count;
};
```

interface

```
int A::count;
A::A() {
   ++count;
}
int A::getCount() {
   return count;
}
```

implementation

```
int main() {
   A a,b,c;
   cout << a.getCount();
}
```

Statikus mezők és tagfüggvények

Statikus tagfüggvény nem hivatkozhat mezőre, kivéve ha az statikus mező

A statikus tagfüggvénynek nincs implicit paramétere, más szavakkal nincs benne this

Ezért olyasmit illik osztályszintű tagfüggvényként megfogalmazni, ami kihasználja, hogy nem kell hozzá objektum, pl graphicslib groutput::instance()

static és öröklődés

- Nem lehet egy tagfüggvény egyszerre virtual és static: az osztályszintűség mindig a statikus típusra vonatkozik
- hasonló eset:
  virtual bool is_focusable() { return true;}
- Szükség van az objektumra, hogy a dinamikus típust megtudjuk.

Hibakezelés

- Hiba felderítése, tesztelés
- Hiba reprodukálhatósága
  - megsejthető a hiba természete
- Hibák nyilvántartása
  - pl. Bugzilla
- Hiba okának felderítése
  - Egyszerű esetben javítás
  - Néha újratervezés

Hibakeresés, tesztelés

- Fekete doboz teszt
- Fehér doboz teszt
  - automatikus forráskódkezelők
  - „test coverage”
- Terhelőtesztek
  - szerver, realtime alkalmazások
  - GUI véletlen eseményekkel

Hibafajták

(A program nem fordul)

A program szabálytalanul terminál („elszáll”)

A program végtelen ciklusba kerül („lefagy”)

A program mást csinál, mint amit a programozó gondol

A program mást csinál, mint amit a felhasználó gondol

bug vs feature

Futásidejű végzetes hibák

- Osztás nullával, moduló nullával
- Idegen memória hivatkozása
  - Mutató nem megfelelő
  - Kimegyünk a vector-ból, string-ből
- Nincs meg a DLL
- Stack overflow: pl. végtelen rekurzió
- Out of memory
- .. egyéb platform és nyelvfüggő esetek

Hibakeresési technikák

- Kód tüzetes vizsgálata
- Favágó módszer: random kikommentezés
- Adatok kiírása a konzolra
- Adatok kiírása fájlba
  - Minden platformon, minden környezetben működik, kivéve a fájlrendszer nélküli mikrokontrolleres eseteket
- Programkövetés, nyomkövetés
  - GDB
  - Szinte minden IDE tartalmaz ilyet