

2D1387 Programsystemkonstruktion med C++

Lösningsförslag, tentamen 15 januari 2004

Obs: Dessa lösningar är just lösningsförslag. Rättningen kan se något annorlunda ut.

Uppgift 1

a) defaultkonstruktör `A()`, kopieringskonstruktör `A(A)`, tilldelningsoperator `oper=(A)` och destruktör `~A()`. Dessutom får man adressoperatören `oper&()` samt (de statiska) allokerings- och avallokeringsoperatorerna `oper new()`, `oper new[]()`, `oper delete()` och `oper delete[]()`.

b)

```
int main()
{
    class A {};
    A *ap = new A;      // A()
    A *aa = new A[3];   // A() x 3
    A a = *ap;          // A(A)
    a = *aa;            // oper=(A)
    delete ap;          // ~A()
    delete [] aa;       // ~A() x 3

    return 0;
}                        // ~A()
```

Uppgift 2

a) Exempel på program:

```
template<class It, class F>
void apply(It beg, It end, F f)
{
    for(; beg != end; ++beg)
        f(*beg);
}
```

b) Exempel på funktion och funktionsobjekt:

```
void add3(int &i) { i += 3; }
struct Add {
    Add(int j) : n(j) {}
    operator()(int &i) { i += n; }
    int n;
};

std::vector<int> v(7);
int a[5] = { 1, 3, 2, 6, 0 };
apply(v.begin(), v.end(), Add(2)); // add 2 to each elem
apply(a, a + 5, add3);             // add 3 to each elem
```

Uppgift 3

```
struct A
{
    static void foo() {}           // 1
    static int bar() { return 0; }
    static const int i = 0;        // 2
    static const int j = 8;        // 3
    int k;                         // 4
    int m;
    int n;

    class B
    {
        B() : j(bar()), n(i) {}    // 6, 7, 8
        int n;
        int j;
    };
    B b;                           // 5
};
```

1: en statisk funktion kan inte vara virtuell. **2:** initieringsvärdet på en statisk konstant måste vara ett konstant värde, returvärdet från en funktion är inte ett konstant värde. **3:** icke-konstanta statiska variabler ska initieras utanför klassen. **4:** medlemsvariabler kan bara initieras i konstruktorn. **5:** klassen B ej deklarerad i sin ursprungliga position. Obs att A har tillgång till Bs privata konstruktor när B är en nästlad klass till A. **6:** A är inte en basklass till B. **7:** A::n är medlemsvariabel och därför krävs ett objekt för åtkomst. **8:** m är inte en medlem i B.

Uppgift 4

a) Typkontroll, minskad risk för syntaxfel (t.ex. `#define LEN 100;`, där semikolon ställer till problem), debuggning (få kompilatorer har bra hantering av preprocessorkonstanter), räckvidd (preprocessorernas konstanter syns i resten av filen).

b) Typkontroll av argument och returvärde, minskad risk för syntaxfel, debuggning, mallar kan specialiseras, funktioner kan överlagras på antal element, makron kan inte konverteras till funktionspekare (och kan därför inte skickas som t.ex. argument till en funktion), argument till makron ger önskat resultat vid blockstatiska variabler, makron kan inte ges defaultvärden.

c) Koden kan inte göras kompatibel med C vid användning av mallar, mallar kan öka kompileringstiden, makron är garanterat `inline`.