

Tentamen i 2D1387 Programsystemkonstruktion med C++

Datum: 17 april 2001

Hjälpmedel: Valfri lärobok i C++

Tid: 5 timmar

Skriv tydligt och ge motiveringar till dina svar. För godkänt krävs 25 poäng för betyg fyra krävs 34 poäng och för betyg fem krävs 40 poäng. Alla betygsgränser är inklusive bonuspoäng. Maxpoäng är 45 utan bonus. Maximalt antal bonuspoäng är nio.

1. Betrakta följande program:

```
struct A {
    A() { foo(); }
    virtual void foo() { cout << "A" << endl; }
};
struct B : public A {
    ~B() { foo(); }
    void foo() { cout << "B" << endl; }
};

int main()
{
    B b;
    b.foo();

    A *ap = new B;
    ap->foo();

    delete ap;
    return 0;
}
```

Vad skriver programmet ut när man kör det? Motivera ditt svar och ge kommentarer för varje rad i main. 7p

2. Mallfunktionen `get_elem` tar som argument en vektor med element av typ T och en jämförelsefunktion f . Funktionaliteten hos `get_elem` förklaras lättast genom ett exempel:

```
bool my_min(int t1, int t2) { return t1 < t2; }

int iv[5] = {1, 3, 6, 2, 4};

int min = get_elem(iv, iv + 5, my_min); // 1
int max = get_elem(iv, iv + 5,
                  std::greater<int>()); // 6

int tmp = get_elem(iv + 2, iv + 5, my_min) + // 2
          get_elem(iv, iv + 2, my_min); // 1
```

I det första anropet använder sig `get_elem` av funktionen `my_min` för att hitta det minsta elementet i vektorn. I det andra exemplet använder vi en binär STL-funktor (eng. *functor*, dvs funktionsobjekt) för att hitta det största elementet.

- a) Vi tänker oss för tillfället att kompilatorn ersätter alla templateparametrar med deras faktiska typer. Ge en deklaration som matchar det första anropet till `get_elem` i exemplet. 4p
- b) Implementera `get_elem` så att den utför det arbete som specificeras ovan och ger de svar som anges i exemplet. 8p

3. Följande program innehåller fyra fel och kan därför inte kompileras. Vilka är felen? Ge förslag på rättningar där detta är tillämpligt. 12p

```
struct A {
    virtual ~A();
    virtual int foo() const = 0 { return 7; }
};
struct B : public A {
    virtual int foo() const { return A::foo(); }
    virtual int bar()      { return 8; }
};
struct C : public B {
    virtual int bar() { return 9; }
};

const A *pa1 = new C;
pa1->foo();

C *pc = new B;
pc->bar();

A *const pa2 = pa1;
pa2->bar();
delete pa2;
```

4. Stacken är ett exempel på lagringstyp (*storage type*). När man skapar ett lokalt objekt säger man att detta skapas på stacken.

- a) Beskriv lagringstyperna i C++ (inklusive stacken). När, var och hur sker allokering och deallokering av objekten? 6p
- b) Du håller på att skriva ett bibliotek för flera trådar (*multi threaded*). Du vill skapa en trådklass `Thread` med följande egenskaper: `Thread` ska när tråden räknat klart **frigöra sig själv** och får därför **inte allokeras på stacken**. Trådens arbete ska kunna **avslutas med ett funktionsanrop** och tråden ska då frigöras. Implementera dessa begränsningar och funktioner i C++.

Tips: Vilka funktioner körs alltid för objekt skapade på stacken? 8p