Test 2 / Teil 3

Test2-Teil3.pptx

Instanzen und Vektor

Im Folgenden sind nicht immer die Lösungen angegeben, aber Sie können im Zweifelsfall selber den Compiler „fragen“.

Test4-Teil3.pdf

```
void pferd7() {
    std::vector<int> vecI;
    for (int i = 0; i < 11; ++i) {
        vecI.push_back(i);
    }
```

Fortsetzung nächste Folie

```
void pferd7() {
    std::vector<int> vecI;
    for (int i = 0; i < 11; ++i) {
        vecI.push_back(i);
    }
```

```
    for (int i = 8; i > 5; --i) {
        datei << vecI[i] << " ";
    }
```

Ausgabe
1. 8 7 6
2. 7 6 5
3. 8 7 6 5
4. 7 6 5 4

```
void pferd7() {
    std::vector<int> vecI;
    for (int i = 0; i < 11; ++i) {
        vecI.push_back(i);
    }
```

```
for (int i = 3; i < 6; i++) {
    datei << vecI.at(i) << " ";
}
```

Ausgabe?

1. 4 5 6
2. 3 4 5 6
3. 3 4
4. 3 4 5

```
class Pferd {
public:
   Pferd(int n = -1) {
      nr = n; datei << "+P:" << n << " "; }
   ~Pferd() {datei << "-P:" << nr << " ";};
private:
   int nr;
};
```

Fortsetzung nächste Folie

```
void pferd8() {
    std::vector<Pferd*> vecPferd;
    for (int i = 2; i < 4; ++i) {
        Pferd* phlp = new Pferd(i);
        vecPferd.push_back(phlp);
    }
    datei << "Ende\n";
}
```

Wie viele Pferde werden erzeugt (Auf dem Stack und auf dem Heap)?

1. 0
2. 1
3. 2
4. 3
5. 4

```
class Pferd {
public:
    Pferd(int n = -1) {
        nr = n; datei << "+P:" << n << " "; }
    ~Pferd() {datei << "-P:" << nr << " ";};
private:
    int nr;
};
```

```
void pferd8() {
    std::vector<Pferd*> vecPferd;
    for (int i = 2; i < 4; ++i) {
        Pferd* phlp = new Pferd(i);
        vecPferd.push_back(phlp);
    }
    datei << "Ende\n";
}
```

Wie viele Pferde werden nach Ausführung der Funktion zerstört, d.h. wie viel Destruktor-Aufrufe?

1. 0
2. 1
3. 2
4. 3
5. 4

```
class Pferd {
public:
    Pferd(int n = -1) {
        nr = n; datei << "+P:" << n << " "; }
    ~Pferd() {datei << "-P:" << nr << " ";};
private:
    int nr;
};
```

```
void pferd8() {
    std::vector<Pferd*> vecPferd;
    for (int i = 2; i < 4; ++i) {
        Pferd* phlp = new Pferd(i);
        vecPferd.push_back(phlp);
    }
    datei << "Ende\n";
}
```

Ausgabe?
1. +P2 +P2 +P3 +P3 Ende
2. +P2 +P3 Ende –P3 – P2
3. +P2 +P3 Ende –P2 – P3
4. +P2 +P3 Ende

```
void pferd9() {
    // mit 5 Pferd initialieren
    std::vector<Pferd> vecPferd(5);
    for (int i = 2; i < 4; ++i) {
        Pferd gaul(i);
        vecPferd[i] = gaul;
        datei << " loop ";
    }
    datei << "Ende\n";
}
```

```
void pferd9() {
    // mit 5 Pferd initialieren
    std::vector<Pferd> vecPferd(5);
```

Ausgabe?
1. keine
2. +P-1 +P-1 +P-1 +P-1 +P-1
3. +P-1 +P0 +P1 +P2 +P3

```
class Pferd {
public:
    Pferd(int n = -1) {
        nr = n; datei << "+P:" << n << " "; }
    ~Pferd() {datei << "-P:" << nr << " ";};
private:
    int nr;
};
```

```
void pferd9() {
   // mit 5 Pferd initialieren
   std::vector<Pferd> vecPferd(5);

   for (int i = 2; i < 4; ++i) {
      Pferd gaul(i);
      vecPferd[i] = gaul;
      datei << " loop ";
   }
```

```
class Pferd {
public:
   Pferd(int n = -1) {
      nr = n; datei << "+P:" << n << " "; }
   ~Pferd() {datei << "-P:" << nr << " ";};
private:
   int nr;
};
```

Ausgabe **nur** in der Schleife

1. keine
2. +P2 +P2 loop –P2 +P3 +P3 loop –P3
3. +P2 loop –P2 +P3 loop –P3
4. +P2 loop +P3 loop –P3 –P2

```
void pferd9() {
   // mit 5 Pferd initialieren
   std::vector<Pferd> vecPferd(5);

   for (int i = 2; i < 4; ++i) {
      Pferd gaul(i);
      vecPferd[i] = gaul;
      datei << " loop ";
   }
```

```
class Pferd {
public:
   Pferd(int n = -1) {
      nr = n; datei << "+P:" << n << " "; }
   ~Pferd() {datei << "-P:" << nr << " ";};
private:
   int nr;
};
```

Ausgabe **nur** in der Schleife

1. keine
2. +P2 +P2 loop –P2 +P3 +P3 loop –P3
3. +P2 loop –P2 +P3 loop –P3
4. +P2 loop +P3 loop –P3 –P2

„vecPferd[i] = gaul;“ ruft den Standard – Zuweisungsoperator von Pferd auf.
Es wird dort zwar ein neues Pferd erzeugt, aber es erfolgt keine Ausgabe, denn der Standardkopierkonstruktor von Pferd gibt nicht aus.
Das Zerstören über vecPferd ist später.
Somit ist die Ausgabe „ +P2 loop –P2 +P3 loop –P3“.

```
void pferd9() {
   // mit 5 Pferd initialieren
   std::vector<Pferd> vecPferd(5);

  for (int i = 2; i < 4; ++i) {
     Pferd gaul(i);
     vecPferd[i] = gaul;
     datei << " loop ";
     }
     datei << "Ende\n";
}
```

Ausgabe nach der Schleife?

1. Ende
2. Ende –P-1 –P-1 –P2 –P3 -P-1
3. Ende –P-1 –P-1 –P-1 –P-1 -P-1
4. Ende –P-1 –P-1 –P-1 –P-1 -P-1 –P2 –P3

```
void pferd9() {
   // mit 5 Pferd initialieren
   std::vector<Pferd> vecPferd(5);

  for (int i = 2; i < 4; ++i) {
     Pferd gaul(i);
     vecPferd[i] = gaul;
     datei << " loop ";
     }
     datei << "Ende\n";
}
```

Ausgabe nach der Schleife?

1. Ende
2. Ende –P-1 –P-1 –P2 –P3 -P-1
3. Ende –P-1 –P-1 –P-1 –P-1 -P-1
4. Ende –P-1 –P-1 –P-1 –P-1 -P-1 –P2 –P3

vecPferd enthält 5 Instanzen von Pferd.
gaul ist bereits (zweimal) in der Schleife (erzeugt und) zerstört worden
Somit werden 5 Destruktoren aufgerufen
Die Ausgabe ist somit „Ende –P-1 –P-1 –P2 –P3 -P-1“.