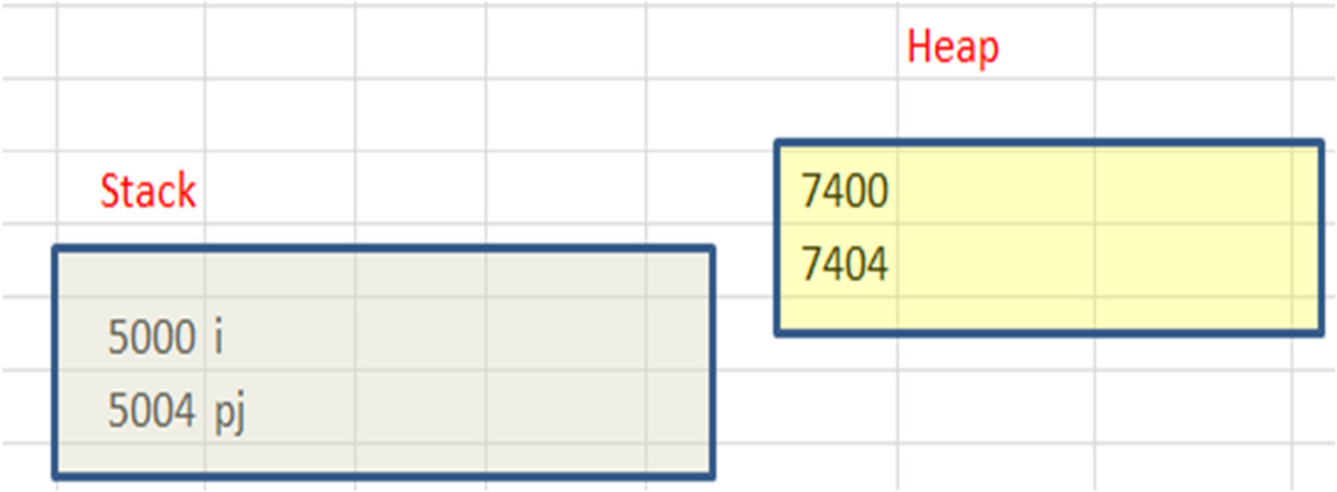
Test 2 / Teil 1

Speicherbelegung auf dem Stack und Heap

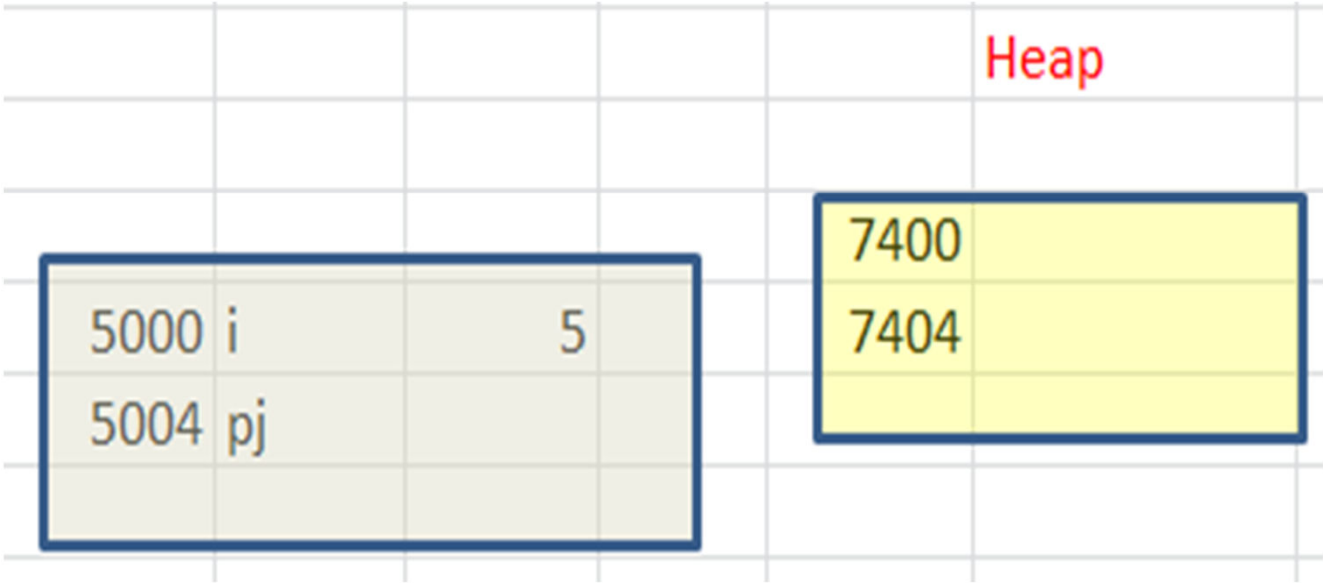
```
void funk14(int* p) {
    p = new int(22); /* 3 */
}
void malen() {
    int i = 5;
    int* pj = &i; /* 1 */
    pj = new int(4);  /* 2 */
    funk14(pj);
    datei << *pj;
}
```

Speicherbelegung zum Zeitpunkt /*2*/



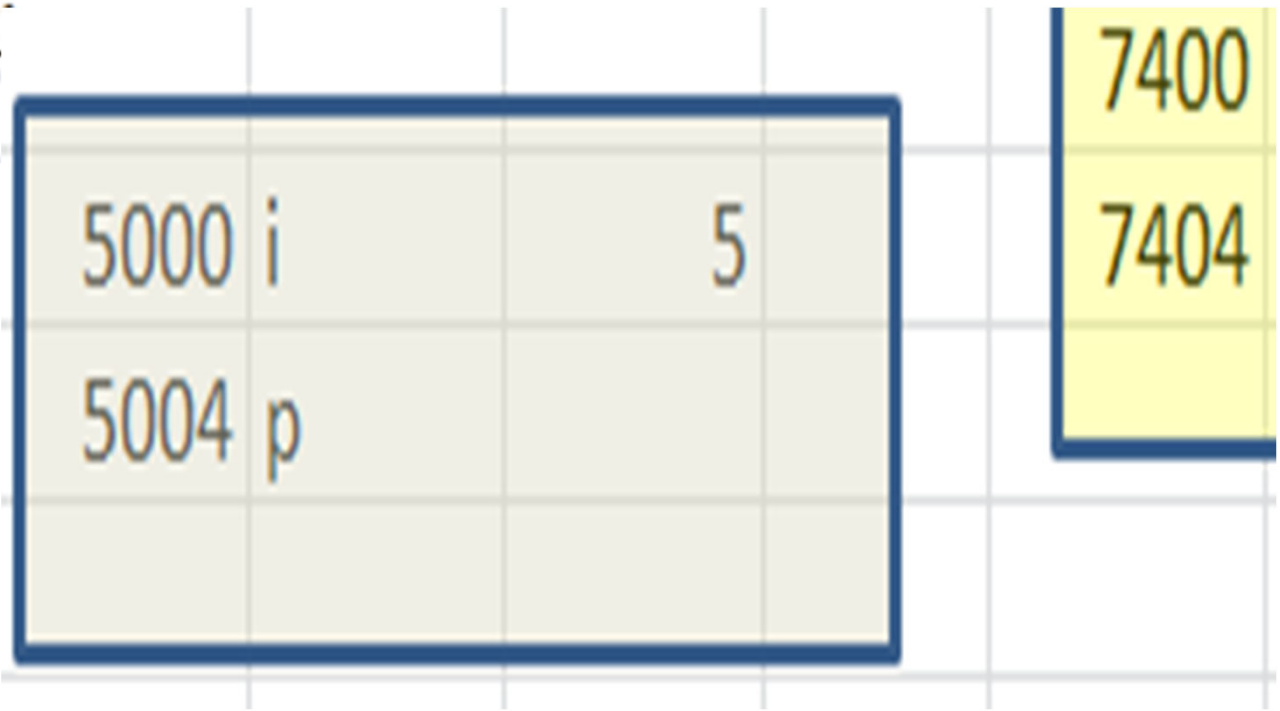
```
void funk14(int* p) {
    p = new int(22); /* 3 */
}
void malen() {
    int i = 5;
    int* pj = &i; /* 1 */
    pj = new int(4);  /* 2 */
    funk14(pj);
    datei << *pj;
}
```

Speicherbelegung zum Zeitpunkt /*3*/



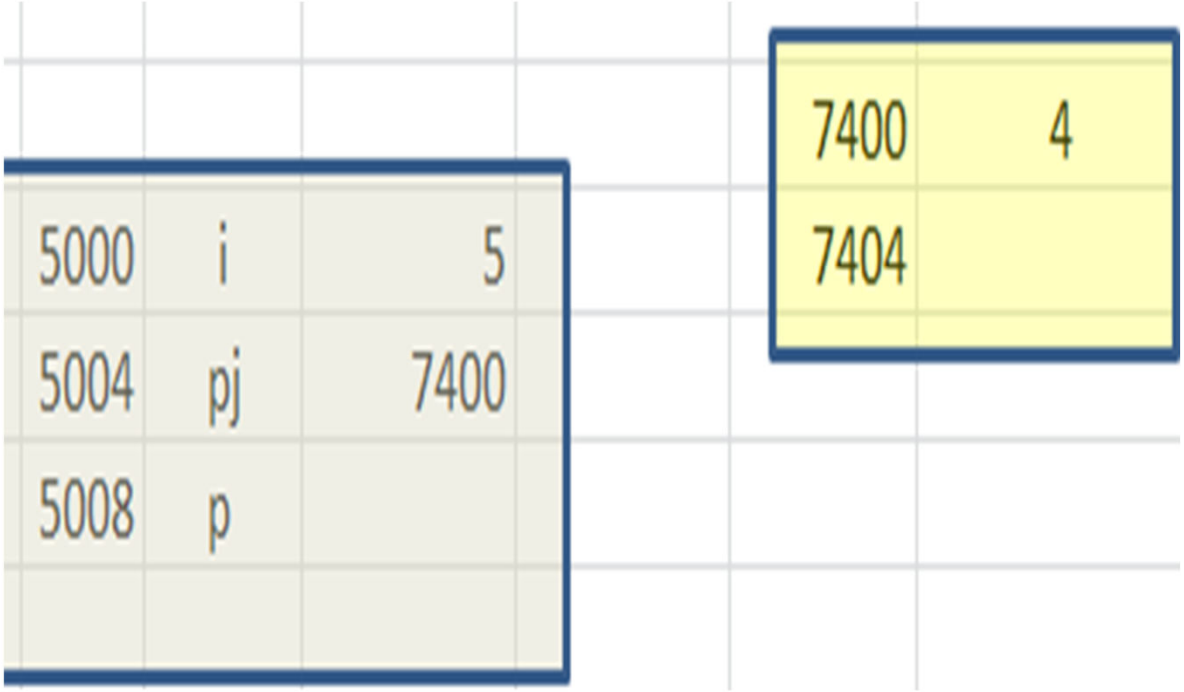
```
void funk14(int p) {
    p = 22;
}
void malen() {
    int i = 5;
    funk14(i);
}
```

Lösung für die Speicherbelegung nach : „int i = 5;“



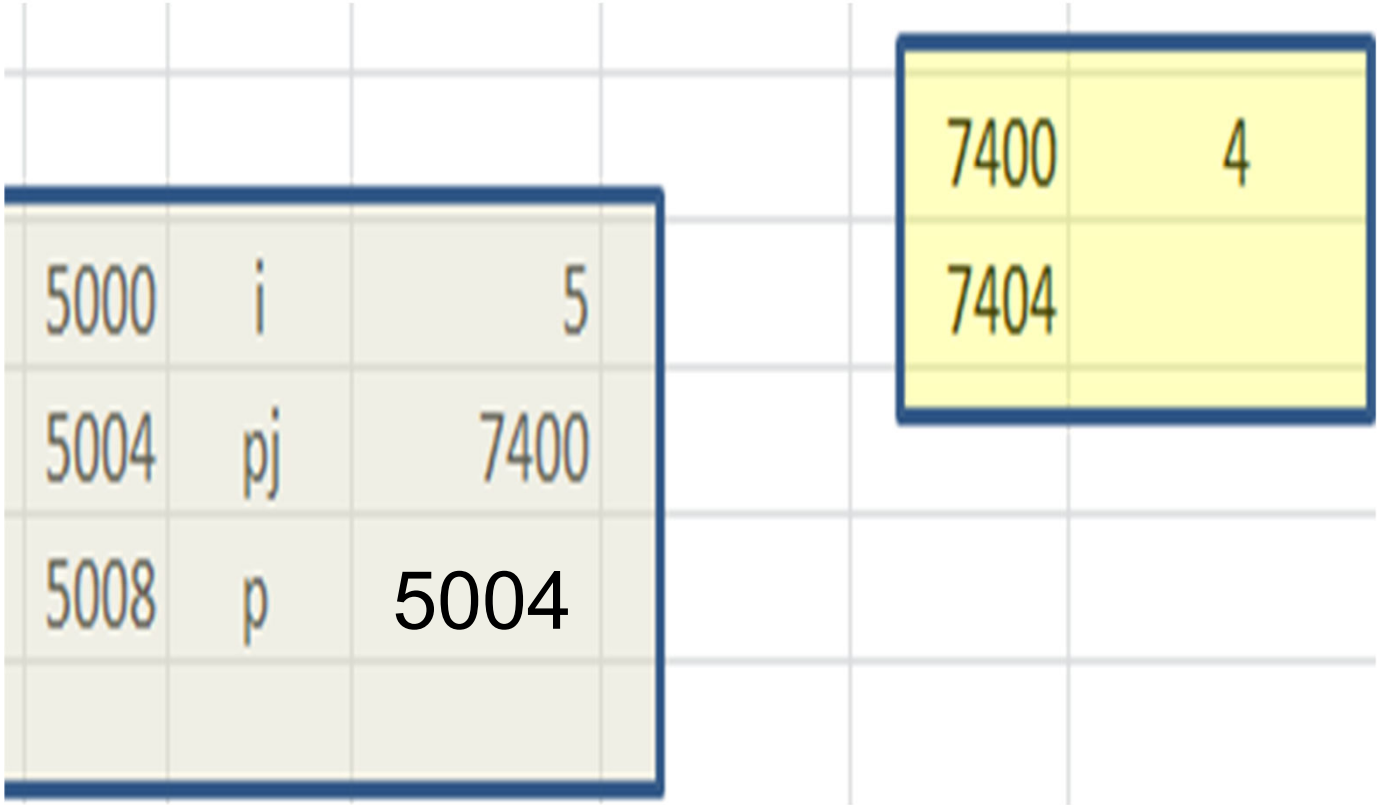
```
void funk14(int* p) {
    p = new int(22); /* 3 */
}
void malen() {
    int i = 5;
    int* pj = &i; /* 1 */
    pj = new int(4);  /* 2 */
    funk14(pj);
    datei << *pj;
}
```

Lösung für die Speicherbelegung
Zeitpunkt /*2*/



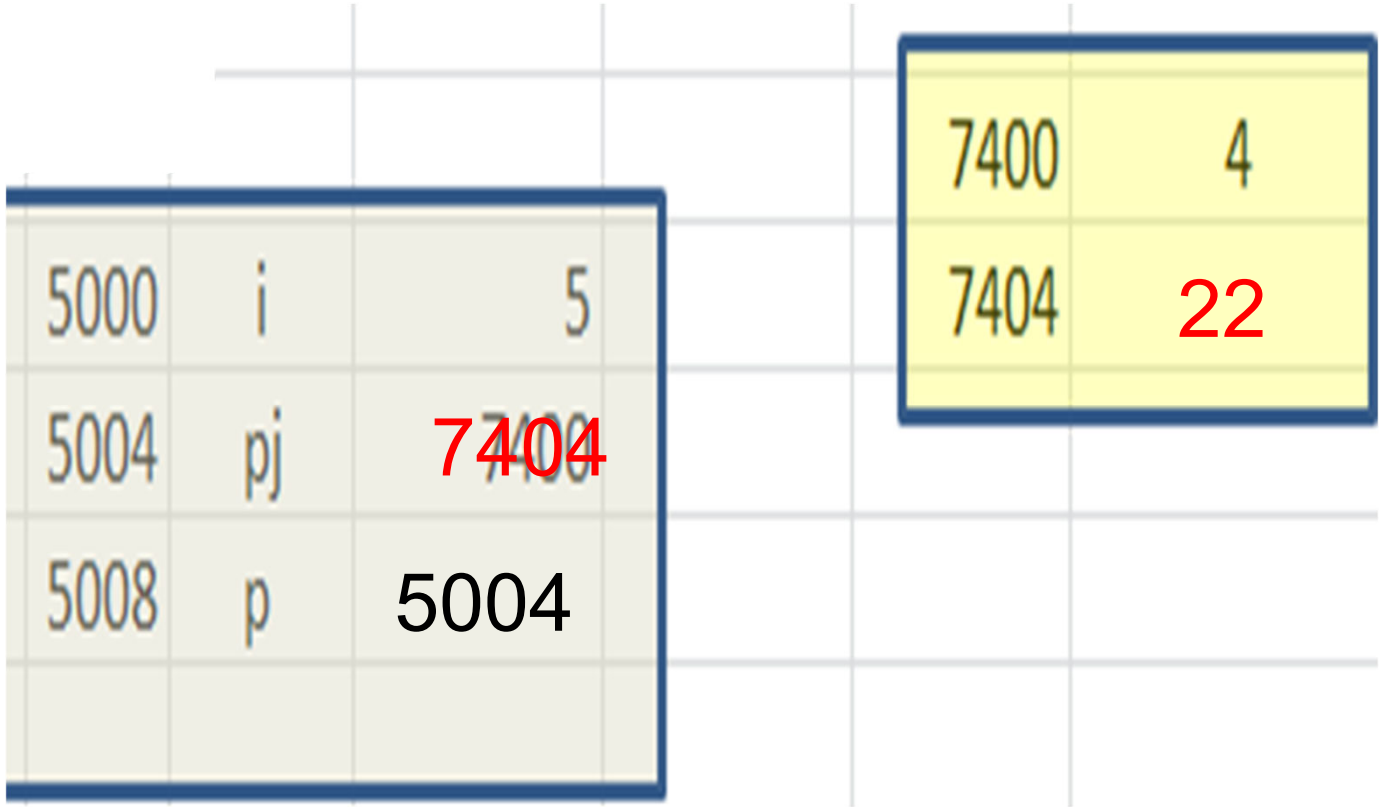
```
void funk14(int*& p) {
    p = new int(22); /* 3 */
}
void malen() {
    int i = 5;
    int* pj = &i; /* 1 */
    pj = new int(4);  /* 2 */
    funk14(pj);
    datei << *pj;
}
```

Lösung für die Speicherbelegung vor dem Zeitpunkt /*3*/



```
void funk14(int*& p) {
    p = new int(22); /* 3 */
}
void malen() {
    int i = 5;
    int* pj = &i; /* 1 */
    pj = new int(4);  /* 2 */
    funk14(pj);
    datei << *pj;
}
```

Lösung für die Speicherbelegung zum Zeitpunkt /*3*/



Test 2 / Teil 2

Objekterzeugung und -zerstörung

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

Clicker-“Abstimmung“

```
class Katze {
public:
    Katze(int n = 0) {
        nr = n; datei << "+K:" << n << " "; }
    Katze(string farbe) {
        nr = 11; datei << "+K:" << nr << " ";}
    ~Katze() {datei << "-K:" << nr << " ";};
    void setNummer(int n) { nr = n; }
private:
    int nr;
};
```

```
void katze1() {
    Katze kazimir(14);
    Katze wolli;
}
```

Welche Ausgabe ergibt sich bei Aufruf von katze1?

1. +K: 14 +K: 0 -K: 0 -K: 14
2. +K: 11 +K: 0 -K: 0 -K: 11
3. +K: 11 +K: 0 -K: 11 -K: 0
4. +K: 14 +K: 11

Speicherbelegung auf Stack und Heap?

```
class Katze {
public:
   Katze(int n = 0) {
      nr = n; datei << "+K:" << n << " "; }
   Katze(string farbe) {
      nr = 11; datei << "+K:" << nr << " ";}
   ~Katze() {datei << "-K:" << nr << " ";};
   void setNummer(int n) { nr = n; }
private:
   int nr;
};
```

```
void katze3() {
    Katze kazimir(14);
    Katze wolli();
    datei << "Ende ";
}
```

double sin(float x);
Katze wolli = Katze();

Welche Ausgabe ergibt sich bei Aufruf von katze3?

1. +K14 +K 11 Ende -K 11 -K 14
2. +K: 14 Ende -K: 14
3. +K: 14 +K: 0 Ende -K: 0 -K: 14
4. +K: 14 Ende +K: 0

```
class Katze {
public:
   Katze(int n = 0) {
      nr = n; datei << "+K:" << n << " "; }
   Katze(string farbe) {
      nr = 11; datei << "+K:" << nr << " ";}
   ~Katze() {datei << "-K:" << nr << " ";};
   void setNummer(int n) { nr = n; }
private:
   int nr;
};
```

```
void katze4() {
   Katze* pk = new Katze("gelb");
   for (int i = 5; i < 8; ++i) {
      pk = new Katze(i);
   }
   datei << "Ende\n";
   delete pk;
}
```

Wie oft läuft die Schleife mit „++i“ durch?

1. 2 mal
2. 3 mal
3. 4 mal
4. 5 mal

```
class Katze {
public:
   Katze(int n = 0) {
      nr = n; datei << "+K:" << n << " "; }
   Katze(string farbe) {
      nr = 11; datei << "+K:" << nr << " ";}
   ~Katze() {datei << "-K:" << nr << " ";};
   void setNummer(int n) { nr = n; }
private:
   int nr;
};
```

```
void katze4() {
   Katze* pk = new Katze("gelb");
   for (int i = 5; i < 8; i++) {
      pk = new Katze(i);
   }
   datei << "Ende\n";
   delete pk;
```

Wie oft läuft die Schleife mit „i++“ durch?

1. 2 mal
2. 3 mal
3. 4 mal
4. 5 mal

```
class Katze {
public:
   Katze(int n = 0) {
      nr = n; datei << "+K:" << n << " "; }
   Katze(string farbe) {
      nr = 11; datei << "+K:" << nr << " ";}
   ~Katze() {datei << "-K:" << nr << " ";};
   void setNummer(int n) { nr = n; }
private:
   int nr;
};
```

```
void katze4() {
   Katze* pk = new Katze("gelb");
   for (int i = 5; i < 8; i++) {
      pk = new Katze(i);
   }
   datei << "Ende\n";
   delete pk;
```

Wie viele Katze-Instanzen werden erzeugt?
Die Schleife selber wird 3mal durchlaufen!!!
1. 2 Katzen
2. 3 Katzen
3. 4 Katzen
4. 5 Katzen

```
class Katze {
public:
   Katze(int n = 0) {
      nr = n; datei << "+K:" << n << " "; }
   Katze(string farbe) {
      nr = 11; datei << "+K:" << nr << " ";}
   ~Katze() {datei << "-K:" << nr << " ";};
   void setNummer(int n) { nr = n; }
private:
   int nr;
};
```

```
void katze4() {
   Katze* pk = new Katze("gelb");
   for (int i = 5; i < 8; i++) {
      pk = new Katze(i);
   }
   datei << "Ende\n";
   delete pk;
```

Wie viele Katze-Instanzen werden zerstört, d.h. wie oft wird der Destruktor für Katze aufgerufen

Die Schleife selber wird 3mal durchlaufen!!!

1. 1mal wird Destruktor für Katze aufgerufen
2. 2mal wird Destruktor für Katze aufgerufen
3. 4mal wird Destruktor für Katze aufgerufen
4. 5mal wird Destruktor für Katze aufgerufen

```
class Katze {
public:
   Katze(int n = 0) {
      nr = n; datei << "+K:" << n << " "; }
   Katze(string farbe) {
      nr = 11; datei << "+K:" << nr << " ";}
   ~Katze() {datei << "-K:" << nr << " ";};
   void setNummer(int n) { nr = n; }
private:
   int nr;
};
```

```
void katze4() {
   Katze* pk = new Katze("gelb");
   for (int i = 5; i < 8; ++i) {
      pk = new Katze(i);
   }
   datei << "Ende\n";
   delete pk;
}
```

Welche Ausgabe ergibt sich bei Aufruf von katze4?

1. +K:11 +K: 5 +K: 6 +K: 7 Ende –K:7 –K:6 –K:5
2. +K:11 +K: 5 +K: 6 +K: 7 Ende –K: 7
3. +K:11 +K: 5 +K: 6 +K: 7 + K: 8 Ende –K: 8
4. +K:0 +K: 5 +K: 6 +K: 7 Ende –K: 0

Speicherbelegung auf Stack und Heap?

Speicherbelegung

```
void katze4() {
    Katze* pk = new Katze("gelb");
    for (int i = 5; i < 8; ++i) {
        pk = new Katze(i);
    }
    datei << "Ende\n";
    delete pk;
}
```

Malen

```
class Katze {
public:
   Katze(int n = 0) {
      nr = n; datei << "+K:" << n << " "; }
   Katze(string farbe) {
      nr = 11; datei << "+K:" << nr << " ";}
   ~Katze() {datei << "-K:" << nr << " ";};
   void setNummer(int n) { nr = n; }
private:
   int nr;
};
```

```
void katze5() {
    Katze* k1[3];              /*1*/
    Katze k2[2];               /*2*/
    k1[1] = &k2[1];            /*3*/
    k1[1]->setNummer(5);       /*4*/
    k1[0] = new Katze(9);      /*5*/
    datei << "Ende ";
}
```

Speicherbelegung auf Stack und Heap mit Papier und Bleistift zu jedem Zeitpunkt

Malen und Überlegen

```
class Katze {
public:
   Katze(int n = 0) {
      nr = n; datei << "+K:" << n << " "; }
   Katze(string farbe) {
      nr = 11; datei << "+K:" << nr << " ";}
   ~Katze() {datei << "-K:" << nr << " ";};
   void setNummer(int n) { nr = n; }
private:
   int nr;
```

```
void streicheln(Katze& k) {
    k.setNummer(-1);
}
void kratzen(Katze k) {
    k.setNummer(21);
}
```

```
void katze6() {
    Katze peter(14);
    streicheln(peter);
    datei << "Peter fertig\n";

    Katze grete(55);
    kratzen(grete);
    datei << "Grete fertig\n";
}
```

Speicherbelegung auf Stack und Heap?

Welche Ausgabe ergibt sich?