Aufgabenblatt 4

- Christian Rebischke (432108)
- Sajedeh Majdi (493981)

Übung 3

Gegeben ist folgendes Programm (nachträglich als `Programm foo` bezeichnet):

```
void foo ( const int ** );
int main()
{
    int ** v = new int * [10];
    foo(v);
    return 0;
}
```

Außerdem gegeben ist folgendes Programm (nachträglich als `Programm bar` bezeichnet):

```
const int * bar ();
int main()
{
    int ** v = new int * [10];
    v[0] = bar();
    return 0;
}
```

Beide Programme gemein haben, dass sie einen Cast versuchen von `int **` zu `const int **`. Auf das `Programm foo` bezogen bedeutet dies, dass der Aufruf `foo(v);` eigentlich ein Argument vom Typ `const int**` haben möchte. `v` ist allerdings vom Typ `int**`. Um genauer zu sein handelt es sich bei `v` um einen Zeiger auf einen Zeiger auf einen Integer. `foo(v);` möchte aber einen Zeiger auf einen Zeiger der wiederum auf einen konstanten Integer zeigt.

Im `Programm bar` ist `bar()` definiert als Funktion, welche einen Zeiger auf einen `const int` Wert zurückgeben soll. Bei `v` handelt es sich aber nur um einen Zeiger auf einen Zeiger der wiederum auf einen Integer zeigt. Dies müsste allerdings `const int` sein damit der Aufruf stattfinden kann.

Die Lösung für das `Programm bar` wäre also entweder aus `int** v = new int *[10]` die Zeile `const int** v = new const int *[10]` zu machen, oder die Definition von `const int * bar();` zu verändern zu `int * bar ();`.

Bei dem `Programm foo` ist die Lösung entweder aus der Definition von `void foo( const int** )` die folgende Definition zu machen: `void foo( int** )` oder besser wäre aus der Definition folgende Definition zu machen: `void foo(const int* const*)`. Bei letzterem wird ein pointer zu einem `const` pointer zu einem `const int` benutzt.

Übung 4

Gegegeben ist folgendes Programm:

```
int foo ( const int & );
int bar ( int & );
int main()
{
    int i = 0;
    int &j = i;
    static const int f = i;
    int * const p = 0;
    p = &i;
    *p = f;
    const int &l = j;
    const int &k = f;
    foo(j);
    bar(l);
    foo(k);
}
```

Bei der Zeile `int * const p = 0;` handelt es sich um eine Initialisierung eines konstanten pointers oder auch read-only pointers. Dadurch wird erreicht, dass die Adresse die im pointer `p` gespeichert wird, nicht mehr verändert werden darf (siehe `p = &i`, diese Operation ist ungültig, da sie die Adresse die in `p` gespeichert wird ändern würde). Die Variable an der Adresse die `p` speichert, darf allerdings weiterhin frei verändert werden (siehe `*p = f;` diese Operation ist demnach legitim, da die Adresse die in `p` gespeichert ist, nicht verändert wird).

Die nächste "Problemzeile" ist `bar(l);`. `bar(l);` ist in so fern ein Problem als, dass die Funktions Signatur (`int bar (int &);`) nicht mit dem Aufruf zusammenpasst. Da `&l` den Typ `const int` hat. Dadurch wird die Funktion nicht gematched und der Aufruf der Funktion ist ungültig. Ein cast von `const int&` auf `int &` ist ebenfalls nicht möglich.