Grundlagen

Aufgabe 1.1 Führen Sie folgendes Programmfragment in Ihrem Kopf aus. Welche Ausgabe würden Sie auf den einzelnen Programmzeilen erwarten?

```
int  a = 0; a = 1; std::cout << a;
int  b = a; a = 2; std::cout << b;
int& c = a; a = 3; std::cout << c;
int& d = c; a = 4; std::cout << d;
```

Aufgabe 1.2 Überlegen Sie sich bei jeder der folgenden Zeilen im Kopf, ob sie zu einem Compilerfehler führt oder nicht. Erklären Sie dann für jede fehlerhafte Zeile, wieso es Sinn macht, dass der Compiler sie nicht akzeptiert.

```
int x; std::cin >> x;

int a = x+1;
const int b = x+2;
constexpr int c = x+3;
a = 1;
b = 2;
c = 3;

int& ra = a;
int& rb = b;
int& rc = c;
ra = 1;
rb = 2;
rc = 3;

const int& cra = a;
const int& crb = b;
const int& crc = c;
cra = 1;
crb = 2;
crc = 3;

constexpr int cca = a;
constexpr int ccb = b;
constexpr int ccc = c;
```

Aufgabe 1.3 Folgende Funktion `f` ermöglicht es, zum Beispiel mit dem Aufruf `f(x)` den Wert x^2 einer gegebenen Variable `x` vom Typ `double` zu berechnen. Mit einem geschachtelten Aufruf wie beispielsweise `f(f(f(x)))` wird sogar der Wert $((x^2)^2)^2 = x^8$ berechnet.

```
double f(double param) {
  const double result = param * param;
  return result;
}
```

Passen Sie die Definition der Funktion `f` so an, dass nach dem blossen Aufruf `f(x)` der berechnete Wert x^2 direkt in der Variable `x` gespeichert wird. Ebenso soll nach dem blossen Aufruf `f(f(f(x)))` der berechnete Wert x^8 in der Variable `x` gespeichert sein.

Um dieses gewünschte Verhalten der Funktion `f` zu erreichen, müssen Sie an verschiedenen Stellen im Code auf geschickte Weise Referenzen einsetzen.