C++ Teil 4

4.1. Deklaration einer Funktion

Bevor eine Funktion benutzt wird, sollte sie immer deklariert werden. Es wird ein sogenannter *Prototyp* erstellt, der vom Compiler auf Richtigkeit überprüft wird.

Die Deklaration sieht der Funktionsdefinition sehr ähnlich. Sie besteht allerdings nur aus dem Kopf, es fehlt der Rumpf (Körper) der Funktion. Die Deklaration wird mit einem Strichpunkt abgeschlossen.

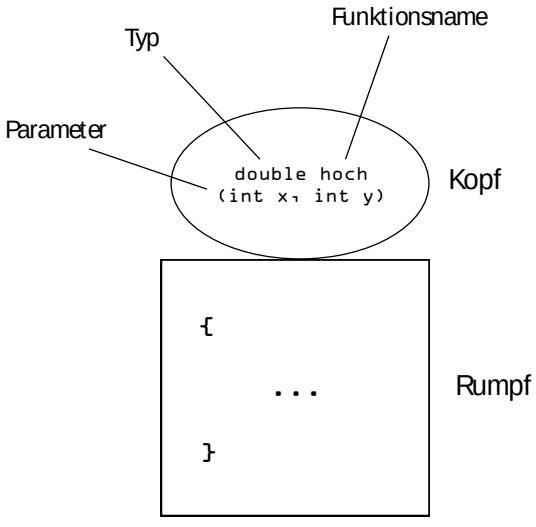
Beispielprogramm

```
// Funktionsdeklaration (Prototyp)
#include <iostream>
using namespace std;

int flaeche(int laenge, int breite); // Prototyp (Deklaration)


int main()
{
	int l, b, f;
	cout << "\nWie breit ist dein Garten? (in m) ";
	cin >> b;
	cout << "\nWie lang ist dein Garten? (in m) ";
	cin >> l;
	f = flaeche(l,b); // Funktion wird aufgerufen
	cout << "\nDein Garten hat eine Flaeche von " << f << " m2";
	return 0;
}


int flaeche(int laenge, int breite) // Funktionsdefinition
{
	return (laenge * breite);
}
```



4.2. Definition einer Funktion

Eine Funktion besteht aus Kopf und Rumpf. Der Typ kennzeichnet den Rückgabewert einer Funktion. Hat die Funktion keinen Rückgabewert, so wird sie mit dem Typ `void` gekennzeichnet. Die Funktion wird mit dem Funktionsnamen von anderen Funktionen aus aufgerufen. Für die Namensgebung gelten die gleichen Regeln wir für Variablennamen. Es werden noch verschiedene

Parameter definiert, die beim Aufruf der Funktion erwartet werden.

Nach dem Funktionskopf folgt der Funktionsrumpf in geschweiften Klammern. Hier findet die eigentliche Implementierung der Funktion statt.

Beispielprogramm

```
// Funktion Potenzieren
#include <iostream>
using namespace std;

double hoch (int x, int y); // Prototyp
int main()
{
	int basis, exponent;
	double ergebnis;
	cout << "Basis: ";
	cin >> basis;
	cout << "\nExponent: ";
	cin >> exponent;
	ergebnis = hoch (basis, exponent);
	cout  <<"\n\n" <<basis<<" hoch "<<exponent<<" = "<<ergebnis;
	return 0;
}
double hoch (int x, int y)
{
	double z = 1.0;
	if (y >= 0) // positiver Exponent
	{
		for ( ; y>0; y--)
		{
			z *= x;
		}
	}
	else // negativer Exponent
	{
		for ( ; y<0; y++)
		{
			z /= x;
		}
	}
	return (z);
}
```

Aufruf

Die Funktion wird nur durch die Angabe des Funktionsnamens aufgerufen, gefolgt von der Liste der übergebenen Parameter. (vgl. obiges Beispiel: `ergebnis = hoch (basis, exponent);`). Der Aufruf kann im Programm überall anstelle eines Ausdrucks stehen. Hat eine Funktion keine Parameter, so bleibt die Liste der Parameter leer. Die Klammern dürfen aber nicht vergessen werden. Beispiel: `int main()`.

Rückgabewert

Eine Funktion wird beendet, wenn die letzte Anweisung der Funktion ausgeführt ist oder wenn sie auf eine `return`-Anweisung trifft (vgl. obiges Beispiel: `return (z);`).

4.3. Rekursion

Wenn eine Funktion sich selbst aufruft, spricht man von **Rekursion**. Dazu ein Beispiel.

Die Fakultät einer Zahl (geschrieben **n!**) ist definiert als $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-2) \cdot (n-1) \cdot n$. Die Fakultät von 5 berechnet sich also folgendermaßen: $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$.

Die Fakultät einer Zahl kann aber auch rekursiv definiert werden: $0! = 1$ und $n! = n \cdot (n-1)!$.
5! berechnet sich mit dieser rekursiven Definition so: $5! = 5 \cdot (5-1)! = 5 \cdot 4!$,
$4! = 4 \cdot (4-1)! = 4 \cdot 3!$, $3! = 3 \cdot (3-1)! = 3 \cdot 2!$, $2! = 2 \cdot (2-1)! = 2 \cdot 1!$, $1! = 1 \cdot (1-1)! = 1 \cdot 0! = 1 \cdot 1$.
Also ist $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$.

Beispielprogramm

```
// Berechnung der Fakultaet mit Hilfe von Rekursion
#include <iostream>
using namespace std;

unsigned long fakultaet(int n); // Prototyp
int main()
{
	int x;
	unsigned long faku;
	cout << "Eingabe einer Zahl: ";
	cin >> x;
	faku = fakultaet(x);
	cout << "\n\n" << x <<"! = " << faku;
	return 0;
}
unsigned long fakultaet(int n)
{
	unsigned long faku;
	if (n==0)
	{
		return 1;
	}
	else
```

```
        {
                faku = n * fakultaet(n-1);
                return (faku);
        }
}
```

Rekursive Funktionen sollten nur sparsam und wohlüberlegt eingesetzt werden. Bei jedem Aufruf wird für jede verwendete Variable in dieser Funktion ein eigener Speicherplatz angelegt, der erst wieder beim Beenden der Funktion freigegeben wird.

Aufgaben

1. Was ist der Unterschied zwischen einer Funktionsdeklaration und einer Funktionsdefinition?
2. Was ist Rekursion?
3. Schreibe einen Prototyp für eine Funktion mit dem Namen `Umfang`, die eine `unsigned long int` Variable zurückgibt und zwei `unsigned short int` Parameter erwartet.
4. Schreibe ein Programm, das zwei Zahlen über die Tastatur einliest und die erste Zahl durch die zweite Zahl dividiert. Eine Funktion soll überprüfen, ob die zweite Zahl = 0 ist. Wenn ja, soll die Funktion den Fehlercode –1 zurückgeben, andernfalls das Divisionsergebnis. Dementsprechend soll entweder eine Fehlermeldung oder das Ergebnis ausgegeben werden.
5. Schreibe ein Programm, das die Fakultät einer Zahl berechnet – und zwar ohne Rekursion.
6. Schreibe ein Programm, das einen Würfel simuliert. Es sollen 1000 Würfe gemacht und am Ende eine Statistik in Form eines Balkendiagramms ausgegeben werden. Du benötigst dazu zusätzlich die beiden Header-Dateien `<stdlib.h>` und `<time.h>`, die wie üblich mittels `#include` in das Programm eingebunden werden. Die Anweisung `int randomize();` initialisiert den Zufallsgenerator von C++. Mit der Anweisung `rand()` wird eine Zufallszahl (random engl. Zufall) berechnet. Um daraus die Zahlen von 1 bis 6 zu erzeugen ist folgende Anweisung nötig: `rand()%6+1;` (**Warum?**).