

# Objektorientiertes Programmieren - SWB2 & TIB2

## Labor 1

### Aufgabe 1: Ausgaben formatieren

Um die Ausgaben über einen Standard-Stream zu formatieren, stehen eine Reihe von Manipulatoren zur Verfügung (siehe die Folien aus der Vorlesung). Während die meisten Manipulatoren in der Bibliothek `iostream` enthalten sind, benötigen die Manipulatoren `setprecision`, `setw` und `setfill` die Bibliothek `iomanip`. Genaue Beschreibungen der Manipulatoren sowie Beispiele finden Sie unter der generell hilfreichen Webadresse <http://www.cplusplus.com/reference>.

Ergänzen Sie das untenstehende Programm an den Stellen `/* xxx */`, so dass es die nachfolgende Ausgabe erzeugt:

Nr	Oct	Hex	String	Fixed	Scientific
1	01	0x1	***** +	1.234	1.234E+000
2	02	0x2	***** -	2.468	-2.468E+000
3	03	0x3	***** +	4.936	4.936E+000
4	04	0x4	***** -	9.872	-9.872E+000
5	05	0x5	***** +	19.744	1.974E+001
6	06	0x6	***** -	39.488	-3.949E+001
7	07	0x7	***** +	78.976	7.898E+001
8	010	0x8	***** -	157.952	-1.580E+002
9	011	0x9	***** +	315.904	3.159E+002
10	012	0xa	***** -	631.808	-6.318E+002

Im folgenden Programm erzeugt der Ausdruck `string(k % 5 + 1, '*')` einen String aus  $k \bmod 5 + 1$  Sternchen.

```
#include <string>
#include /* xxx */
#include /* xxx */
using namespace std;

int main() {
    double d = 1.234;
    cout << "Nr  Oct  Hex String  Fixed      Scientific " << endl;
    for (int k = 1; k < 11; k++) {
        cout << /* xxx */ << k;
        cout << /* xxx */ << k;
        cout << /* xxx */ << k << " ";
        cout << /* xxx */ << string(k % 5 + 1, '*') << " ";
        cout << /* xxx */ << d;
        cout << /* xxx */ << d;
        d = d*-2;
        cout << endl;
    }
```

```
    return 0;
}
```

## Aufgabe 2: Daten einlesen und ausgeben

Schreiben Sie ein Programm, das den Nutzer fragt, wie viele ganze Zahlen eingelesen werden sollen. Lesen Sie dann die gewünschte Anzahl von ganzen Zahlen in ein dynamisches Array ein. Geben Sie alle Zahlen im Array sowie deren Durchschnitt aus. Für die Ein- und Ausgabe nutzen Sie die Streams `cin` und `cout`.

## Aufgabe 3: Ein einfaches Pac-Man-Spiel - Teil 1

Sie werden im nächsten Labor ein kleines Pac-Man-Spiel programmieren. Bei diesem Spiel laufen Geister durch ein Labyrinth und versuchen, Sie zu fangen. Sie dürfen sich nicht fangen lassen und müssen auf dem Weg durch das Labyrinth so viele Münzen auflesen, wie möglich. Das Spiel endet, wenn alle Münzen von Ihnen aufgelesen wurden oder ein Geist Sie erwischt hat.

Als erstes gilt es, ein Labyrinth zu entwerfen. Das könnte man manuell in einem Texteditor tun. Dies ist jedoch langweilig. Daher schreiben wir hier ein Programm, mit dem wir ein Labyrinth „zeichnen“ können. Zuerst sind überall im Labyrinth Mauern. Dann läuft man herum und ersetzt die Mauern durch den Weg, den man gelaufen ist. Ergänzen Sie daher das folgende Programmgerippe an den Stellen `/*HIER */`, um erst einmal ein Labyrinth zu erzeugen.

```
#include <iostream>
#include <conio.h> // für _getch()
// Achtung: _getch() ist nicht im Standard und
//          daher abhängig vom Compiler
using namespace std;

// Größe des Labyrinths
const int kZeilen = 11;
const int kSpalten = 11;

// Zeichen, die im Labyrinth vorkommen können,
// NL = new line, EOS = end of string
enum Symbole { MAUER = '#', WEG = ' ', MUENZE = ':', NL = '\n',
               EOS = '\0', ICH = 'X', GEIST = 'G' };

// Das Labyrinth als char-Array
// Die vorletzte Spalte speichert ein Zeilenendezeichen \n
// und die letzte ein \0-Zeichen, damit die Ausgabe leichter wird.
char labyrinth[kZeilen][kSpalten + 2];

// Labyrinth mit # füllen
void initialisieren() {
    /* HIER */
}

// Labyrinth auf dem Bildschirm ausgeben
```

```

void drucken() {
    // Console frei machen
    system("cls");
    // Labyrinth ausgeben
    /* HIER */
}

// Hilfsfunktion max
int max(int x, int y) {
    return (x <= y) ? y : x;
}

// Hilfsfunktion min
int min(int x, int y) {
    return (x <= y) ? x : y;
}

// Durch Herumlaufen werden Wege im Labyrinth erzeugt
// ASCII-Wert der Tasten: oben 72, links 75, rechts 77, unten 80
void erzeugen() {
    char c = 'x';
    int posx = kSpalten / 2;
    int posy = kZeilen / 2;
    labyrinth[posy][posx] = ICH;
    drucken();
    while (c != 'q') {
        drucken();
        cout << "Laufen mit Pfeiltasten. Beenden mit q." << endl;
        labyrinth[posy][posx] = WEG;
        c = _getch();
        switch (int(c)) {
            // oben
            case 72: posy = max(1, posy - 1); break;
            // links
            case 75: posx = max(1, posx - 1); break;
            // rechts
            case 77: posx = min(kSpalten - 2, posx + 1); break;
            // unten
            case 80: posy = min(kZeilen - 2, posy + 1); break;
            // q = quit
            case 113: break;
        }
        labyrinth[posy][posx] = ICH;
    }
}

int main() {

```

```
    initialisieren();  
    drucken();  
    erzeugen();  
}
```

## Aufgabe 4: Ein- und Ausgabe und Aufzählungstypen

Tsching-Tschang-Tschong, auch bekannt als Schnick-Schnack-Schnuck und das Stein-Schere-Papier-Spiel, kann auch am Computer gespielt werden. Schreiben Sie ein Programm, so dass Sie mit dem Computer Tsching-Tschang-Tschong spielen können. Nutzen Sie das untenstehende Programmgerippe.

```
/*  
  
Spiel Tsching, Tschang, Tschong  
  
Regeln:  
Es gibt zwei Spieler: der Programmierer und der Computer.  
Beide Spieler waehlen gleichzeitig eines der drei Objekte  
Stein, Schere und Papier. Beide können das gleiche Objekt wählen.  
Der Spieler, der das mächtigere Objekt gewählt hat gewinnt.  
SCHERE kann einen STEIN nicht zerschneiden, d.h. STEIN ist  
mächtiger als SCHERE. PAPIER wickelt STEIN ein, d.h. PAPIER ist  
mächtiger als STEIN. SCHERE zerschneidet PAPIER, d. h. SCHERE  
ist maechtiger als PAPIER  
  
*/  
  
#include <cstdlib>  
#include <cstring>  
#include <ctime>  
#include <iostream>  
using namespace std;  
  
// Aufzählungstyp für Stein etc.  
enum objectType { STEIN, SCHERE, PAPIER };  
  
// Struktur für einen Spieler bestehend aus Name und Wahl  
// des Spielers  
struct player {  
    char * name;  
    objectType choice;  
};  
  
// Variable für den Namen des Spielers  
char name[15];  
  
// Name des Spielers eingeben  
char * enterName(char str[])
```

```

{
}

// Den Computer zufällig waehlen lassen.
// Nutzen Sie srand(...) und rand().
objectType randomChoice()
{
}

// Die Wahl von STEIN etc. als String zurückgeben lassen
char * object2str(objectType o)
{
}

// Einen Text mit dem Namen des Spielers und seiner Wahl ausgeben
void showPlayer(player p)
{
}

// Die Wahl des Spielers abfragen
objectType enterChoice()
{
}

// Die Wahl bestimmen, die gewonnen hat
objectType winningObject(objectType obj1, objectType obj2)
{
}

// Ausgeben, wer gewonnen hat
void showWinner(player p1, player p2)
{
}

int main()
{
    player player1, player2;
    player1.name = "Computer";
    player1.choice = randomChoice();
    cout << "Der Computer hat sein Wahl getroffen." << endl;
    player2.name = enterName(name);
}

```

```
player2.choice = enterChoice();
showPlayer(player1);
showPlayer(player2);
showWinner(player1, player2);
}
```

Die Ausgabe des Programms soll folgendermaßen aussehen.

Der Computer hat sein Wahl getroffen.

Name des Spielers:

Bitte Objektwahl eingeben (1 = Stein, 2 = Schere, 3 = Papier):

Computer hat das Objekt Stein gewaehlt.

Daniel hat das Objekt Schere gewaehlt.

Computer hat gewonnen!

### Aufgabe 5: Funktionen mit Defaultwerten

Schreiben Sie eine Funktion `trapezFlaeche(...)`, die die Fläche eines Trapezes berechnet und auf dem Bildschirm ausgibt. Mit den Bezeichnungen der Seiten und Winkel wie in der Abbildung berechnet sich die Fläche mit  $\frac{1}{2} \cdot (a + c) \cdot b \cdot \sin(\gamma)$ . Die Parameter der Funktion sollen die Länge der beiden parallelen Seiten, die Länge der Seite  $b$  und den Winkel  $\gamma$  angeben. Schreiben Sie die Funktion so, dass Sie für

- ein Quadrat nur die Seitenlänge der Seite  $a$ ,
- ein Rechteck nur die Seitenlängen  $a$  und  $b$
- ein Parallelogramm nur die Seitenlängen  $a$  und  $b$  und den Winkel  $\gamma$

angeben müssen. Testen Sie die Funktion in einem kleinen Hauptprogramm mit den folgenden Werten:

- a)  $a = 2.0$ ,  $c = 3.5$ ,  $b = 4.0$ ,  $\gamma = 45.0$
- b)  $a = 3.0$ ,  $b = 4.0$ ,  $\gamma = 45.0$
- c)  $a = 4.0$ ,  $b = 5.0$
- d)  $a = 2.0$

Hinweis: Die Funktion `sin(.)` findet sich in der Bibliothek `cmath`.

