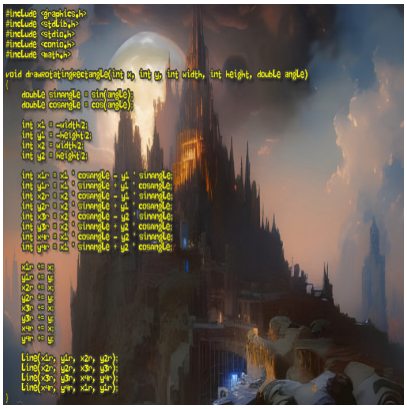


Archaische Programmierung mit Turbo C

### Description

Moderne Entwicklungsumgebungen und Programmiersprachen haben tolle Features. Doch manchmal kann das ganz schön nerven und es ist angenehm, in der Programmierung eine kleine Zeitreise zu machen. Wer Lust hat, kann dies mit Turbo C und DOSBox selbst tun.

## Warum unter DOS?

MS-DOS ist ein altes 16-Bit-Betriebssystem. Das Original wird schon seit vielen Jahren nicht mehr weiterentwickelt, aber es gibt zehntausende Programme und Spiele dafür. Mit DOSBox lässt es sich sehr leicht emulieren. Wer sich ein wenig mit der Konfiguration befasst, kann sich in dieser virtuellen Welt total austoben.

Kurz gesagt: Es ist sehr alt, gibt einem das „back to the roots” Gefühl und es ist vergleichsweise unkompliziert.

## Warum C?

Die Programmiersprache C gibt es seit 1972 und wird heute noch, wenn auch nicht so häufig, verwendet. Es ist eine sehr hardwarenahe Sprache, die aber nicht sonderlich schwer zu erlernen ist. Am C-Syntax orientieren sich viele nachfolgende Sprachen, etwa C++, C#, Java, JavaScript, PHP, GML und andere. D. h. man hat ein gutes Grundverständnis für viele andere Sprachen, wenn man C beherrscht.

Apropos Verständnis: Wenn sich in alten Entwicklungsumgebungen befindet, die letztlich nicht viel mehr waren als ein Texteditor, lernt man m. M. n. wesentlich besser den Umgang mit Code. Es gibt keine Autovervollständigung, keinen Syntax Highlighter, keine zusätzlichen Fenster oder sonstigen Firlefanz. Man konzentriert sich, ohne Ablenkung, nur auf den eigenen Code. Ob der richtig ist, sagt einem später der Compiler.

Im Beispiel verwende ich Turbo C 2 von 1989. Man macht alles darin mit der Tastatur. Am Anfang fühlt

sich das sehr ungewohnt an, aber mit jedem kleinen Programm macht das immer mehr Spaß und der Fokus liegt auf der Programmierung.

# Vorbereitung

Um loslegen zu können, braucht es drei Sachen:

1. DOSBox
2. Turbo C
3. Zeit

Für die DOSBox gibt es unzählige Anleitungen. Deshalb will ich diesbezüglich nur die wichtigsten Schritte nennen.

# DOSBox

Aktuell ist die Version 0.74-3. Einfach die Version für euer Betriebssystem herunterladen und ggf. installieren. Wenn ihr Windows habt, schaut im Ordner nach, in welches DOSBox installiert wurde: da müsste die Datei „DOSBox 0.74-3 Options.bat" liegen. Diese müsst ihr starten, um DOSBox zu konfigurieren. In anderen Betriebssystemen müsst ihr die entsprechende *.conf*-Datei suchen. Hier ein paar meiner wichtigsten Einstellungen, die ihr entsprechend bei euch ändern bzw. ergänzen solltet:

```
[sdl]
fullscreen=true
fulldouble=true
fullresolution=desktop
windowresolution=1920x1080
output=ddraw

[dosbox]
machine=svga_s3
memsize=16

[render]
frameskip=0
aspect=false
scaler=normal3x

[cpu]
core=dynamic
cputype=pentium_slow
cycles=16000
cycleup=500
cycledown=500

[mixer]
nosound=false
rate=49716
blocksize=8192
prebuffer=8192
```

```
[sblaster]
sbtype=sb16
sbbase=220
irq=7
dma=1
hdma=5
sbmixer=true
oplmode=opl3
oplemu=compat
oplrate=49716

[gus]
gus=true
gusrate=49716
gusbase=240
gusirq=5
gusdma=3
ultradir=C:\ULTRASND

[speaker]
pcspeaker=true
pcrate=49716
tandy=auto
tandyrate=49716
disney=true

[dos]
xms=true
ems=emm386
vcpi=true
emm386 startup active=true
umb=true
ver= 6.22
keyboardlayout=auto

[ipx]
ipx=false

[autoexec]
keyb de
mount c e:\DOS\
c:
cls
```

Die meisten Sachen braucht man nicht für Turbo C, aber so habt ihr das schon recht gut konfiguriert, falls ihr noch ein wenig zocken wollt.

**Wichtig:** die drittletzte Zeile beachten! Hier bestimmt ihr, welcher Ordner eure virtuelle Festplatte für DOSBox ist. Bei mir ist das *e:\dos*\. Hier müsst ihr dann alles (am besten mit Unterordnern) ablegen, was ihr in der DOSBox ausführen wollt.

# Turbo C 2.01

Nun zur Entwicklungsumgebung. Einfach auf die oben verlinkte Seite. Rechts im Fenster „DOWNLOAD OPTIONS" die ZIP herunterladen. In der ZIP findet ihr den Ordner „TC". Diesen kopiert ihr in euer DOS-Verzeichnis.

Wer möchte, kann sich auch die SAMPLES rüber kopieren, für dieses kleine Tutorial wird es aber nicht benötigt.

Nun startet ihr die DOSBox. Wenn ihr auf *C:* seid, gebt ihr *cd tc* ein, um im Ordner von Turbo C zu landen. Anschließend müsst ihr nur noch *tc* eingeben.

Alternativ könnt ihr auch Borland Turbo C++ 3.0 verwenden. Das ist etwas komfortabler, bietet bspw. Mausunterstützung. Im verlinkten Download befindet sich eine EXE Datei, die ihr in einen Ordner mit dem Namen „TC" kopiert und startet. Dann wird alles entpackt. Dateien auf Nachfrage einfach überschreiben.

**Wichtig:** Um die Beispiele unter Borland Turbo C++ 3.0 ausführen zu können, musst ihr die Dateien ZWINGEND als *.c* und nicht als *.cpp* abspeichern! Außerdem gab es Probleme mit den Zeichen *[*, *]*, *{* und *}*, wenn man die deutsche Tastatureinstellung hat. Entweder stellt man in der DOSBox-Config die Tastatur auf *us*, oder man verwendet folgende Tastenkombinationen:

- Alt+91 = [
- Alt+93 = ]
- Alt+123 = {
- Alt+125 = }

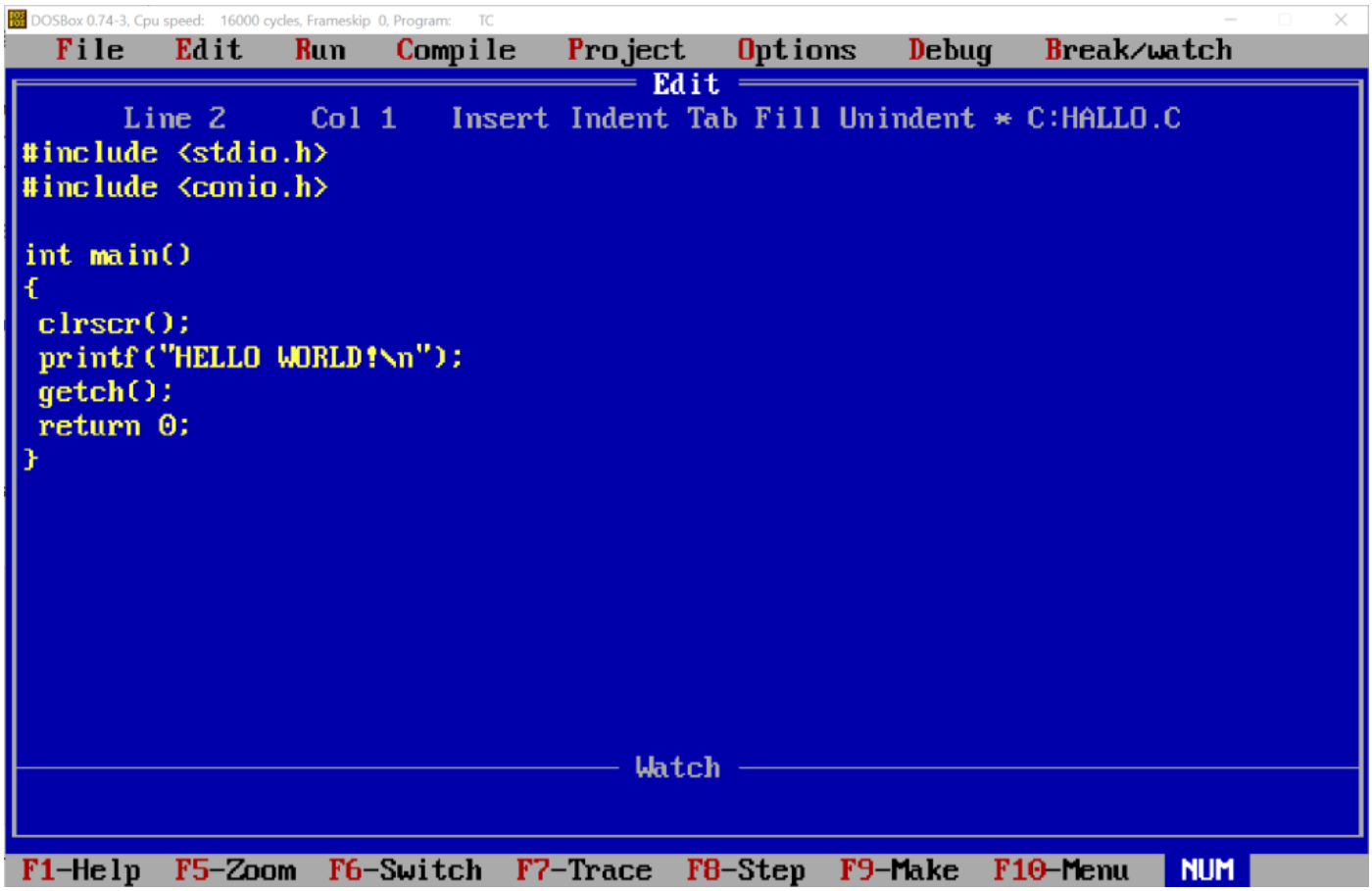
# Hallo Welt

Es ist ratsam, das Programm mit *F2* zu speichern. Die Endung sollte *.c* sein, also geben wir *hallo.c* ein.

```
#include <stdio.h>
#include <conio.h>

int main()
{
 clrscr();
 printf("HELLO WORLD!\n");
 getch();
 return 0;
}
```

Nun speichern und ausführen. Ausführen kann man es mit *Strg+R* und dann auf *Run*.



Hallo Welt in Turbo C

Doch was haben wir nun getan? Schauen wir es uns in kleinen Abschnitten an.

```
int main()
```

Hier beginnt die `main`-Funktion, die in jedem C-Programm benötigt wird und den Startpunkt des Programms darstellt.

```
printf("HELLO WORLD!\n");
```

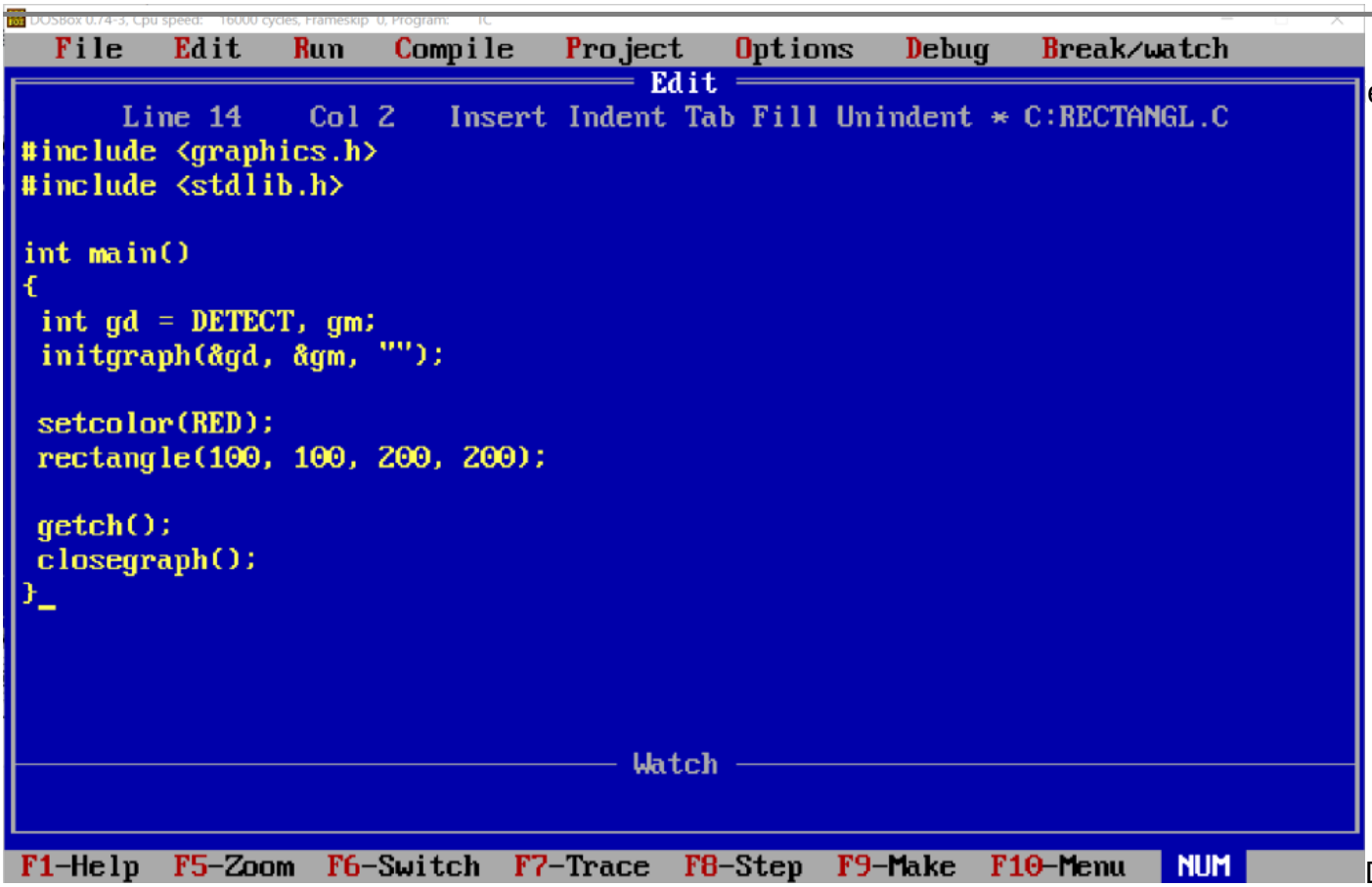
Die Funktion `printf` aus der Header-Datei `<stdio.h>` gibt eine formatierte Zeichenkette auf der Konsole aus. Hier wird die Zeichenkette `"HELLO WORLD!\n"` mit einem Zeilenumbruch am Ende (\n) ausgegeben.

```
return 0;
```

Die Funktion `main` gibt den Wert `0` zurück, um anzuzeigen, dass das Programm erfolgreich ausgeführt wurde.

## Grafikmodus und Rechteck zeichnen

Das war schon ganz nett, aber sicher nicht beeindruckend. Also schalten wir in den Grafikmodus um und zeichnen ein rotes Rechteck.




Der Code ist ebenfalls recht übersichtlich:

```
#include <graphics.h>
#include <stdlib.h>

int main()
{
 int gd = DETECT, gm;
 initgraph(&gd, &gm, „");

 setcolor(RED);
 rectangle(100, 100, 200, 200);

 getch();
 closegraph();
}
```

Nun schauen wir es uns genauer an.

```
int main()
```

Wir rufen wieder das Hauptprogramm auf. Manchmal findet man im Internet auch `void main()`. Ich muss anmerken, dass das in C kein Standard ist und es normalerweise als fehlerhaft angesehen wird.

Das korrekte Hauptprogramm sollte immer `int main()` lauten. Die Verwendung von `void main()` kann in einigen Fällen dazu führen, dass das Programm auf bestimmten Plattformen oder unter bestimmten Bedingungen nicht korrekt funktioniert.

```
initgraph(&gd, &gm, „");
```

`initgraph` initialisiert die Grafiktreiber und legt die Grafikumgebung fest. Die Funktion hat drei Parameter:

- `gd` ist ein Pointer auf eine Variable, die den Grafiktreiber identifiziert, den du verwenden möchtest.
- `gm` ist ein Integer, der die Auflösung und die Farbtiefe der Grafik definiert.
- `""` ist ein leerer String und wird verwendet, um das Fenster zu benennen, in dem die Grafik gezeichnet wird.

In diesem Fall ist `gd` der Wert `DETECT`, was bedeutet, dass die Bibliothek versucht, den Grafiktreiber automatisch zu erkennen. `gm` ist zunächst nicht initialisiert, wird aber später auf einen bestimmten Wert gesetzt. Der leere String gibt an, dass das Fenster keinen Namen haben soll.

```
getch();
closegraph();
```

`getch()` wartet auf eine Eingabe von der Tastatur, bevor das Programm beendet wird. Es ist sozusagen eine Pause am Ende des Programms, damit der Benutzer Zeit hat, das Ergebnis des Programms zu betrachten.

`closegraph()` beendet die Grafikbibliothek und gibt den Speicher frei, der von ihr verwendet wurde. Es wird normalerweise am Ende des Programms aufgerufen, um sicherzustellen, dass die Grafikumgebung sauber geschlossen wird.

# Drehendes Rechteck

Wie man sehen kann, machen wir in C viele Dinge selbst, die uns bspw. in GML abgenommen werden. Das ist hilfreich, um zu verstehen, was im Hintergrund tatsächlich passiert. Nun wollen wir ein drehendes Rechteck anzeigen lassen.

## Theorie

Wenn wir *rectangle* verwenden, übergeben wir vier Argumente. Das Problem ist, dass sie nur vertikale und horizontale Linien zeichnet. Wenn wir es im Kreis drehen lassen wollen, funktioniert das nicht. Wir müssen also vier Linien einzeln zeichnen und deren Winkel sowie Positionen neu berechnen.

Kurz: Hier geht es ans Eingemachte. Die Mathematik dahinter erspare ich euch bzw. würde dieses Tutorial etwas sprengen.

Die Berechnung der Linien lagern wir aus. In einer Endlosschleife lassen wir anschließend das Rechteck, welches nun aus vier einzelnen Linien besteht, drehen.

## Code

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>

void drawRotatingRectangle(int x, int y, int width, int height, double angle)
{
    double sinAngle = sin(angle);
    double cosAngle = cos(angle);

    int x1 = -width/2;
    int y1 = -height/2;
    int x2 = width/2;
    int y2 = height/2;

    int x1r = x1 * cosAngle - y1 * sinAngle;
    int y1r = x1 * sinAngle + y1 * cosAngle;
    int x2r = x2 * cosAngle - y1 * sinAngle;
    int y2r = x2 * sinAngle + y1 * cosAngle;
    int x3r = x2 * cosAngle - y2 * sinAngle;
    int y3r = x2 * sinAngle + y2 * cosAngle;
    int x4r = x1 * cosAngle - y2 * sinAngle;
    int y4r = x1 * sinAngle + y2 * cosAngle;

    x1r += x;
    y1r += y;
    x2r += x;
    y2r += y;
    x3r += x;
    y3r += y;
    x4r += x;
    y4r += y;

    line(x1r, y1r, x2r, y2r);
    line(x2r, y2r, x3r, y3r);
    line(x3r, y3r, x4r, y4r);
    line(x4r, y4r, x1r, y1r);
}

int main()
{
 int gd = DETECT, gm;
 int x = 320;
 int y = 200;
 int width = 200;
 int height = 200;
 double angle = 0.0;

 initgraph(&gd, &gm, „");
```

```
 setcolor(RED);

 while (!kbhit())
 {
   cleardevice();

   drawRotatingRectangle(x, y, width, height, angle);
   angle += 0.1;
   delay(160);
 }

 closegraph();
 return 0;
}
```

Wir stellen gleich mehrere Sachen fest:

1. Es ist deutlich mehr Code
2. Wir brauchen im Header mehr Dinge
3. Wir haben neben der Main-Funktion noch drawRotatingRectangle
4. Wir erkennen eine While-Schleife

## Erklärungen

### Header

```
void drawRotatingRectangle(int x, int y, int width, int height, double angle)
{
    double sinAngle = sin(angle);
    double cosAngle = cos(angle);

    int x1 = -width/2;
    int y1 = -height/2;
    int x2 = width/2;
    int y2 = height/2;

    int x1r = x1 * cosAngle - y1 * sinAngle;
    int y1r = x1 * sinAngle + y1 * cosAngle;
    int x2r = x2 * cosAngle - y1 * sinAngle;
    int y2r = x2 * sinAngle + y1 * cosAngle;
    int x3r = x2 * cosAngle - y2 * sinAngle;
    int y3r = x2 * sinAngle + y2 * cosAngle;
    int x4r = x1 * cosAngle - y2 * sinAngle;
    int y4r = x1 * sinAngle + y2 * cosAngle;

    x1r += x;
    y1r += y;
    x2r += x;
    y2r += y;
    x3r += x;
    y3r += y;
    x4r += x;
    y4r += y;
```

```
    line(x1r, y1r, x2r, y2r);
    line(x2r, y2r, x3r, y3r);
    line(x3r, y3r, x4r, y4r);
    line(x4r, y4r, x1r, y1r);
}
```

Diese Funktion zeichnet ein rotierendes Rechteck mit einer gegebenen Breite, Höhe und Winkel. Die Parameter *x* und *y* geben die Koordinaten des Mittelpunkts des Rechtecks an, während der Parameter *angle* den Winkel der Rotation in Radiant angibt.

Die Funktion verwendet die trigonometrischen Funktionen *sin* und *cos*, um die Koordinaten der Eckpunkte des ursprünglichen Rechtecks auf ihre neuen Positionen nach der Rotation zu transformieren. Die vier Eckpunkte werden dann um die Mittelpunktskoordinaten verschoben und durch vier Linien verbunden, um das rotierende Rechteck zu zeichnen.

Die Funktion arbeitet wie folgt:

- Es werden die Sinus- und Cosinuswerte des Winkels berechnet.
- Es werden die Koordinaten der vier Eckpunkte des ursprünglichen Rechtecks relativ zum Mittelpunkt berechnet.
- Jeder Eckpunkt wird um den Mittelpunkt verschoben, um die tatsächlichen Koordinaten zu erhalten.
- Die vier Eckpunkte werden durch Linien verbunden, um das rotierende Rechteck zu zeichnen.

**main()**

1. Die main-Funktion wird definiert.
2. Zunächst werden zwei Variablen *gd* und *gm* definiert, wobei *gd* mit *DETECT* initialisiert wird.
3. Es werden vier weitere Variablen definiert: *x*, *y*, *width* und *height*, die die Koordinaten und Abmessungen des Rechtecks darstellen, das gezeic
   *angle* wird ebenfalls definiert und auf 0.0 gesetzt, was die Startposition des R
4. Mit *initgraph()* wird das Grafiksystem initialisiert.
5. Mit *setcolor()* wird die Farbe des Rechtecks auf Rot gesetzt.
6. Die **while**-Schleife wird ausgeführt, solange keine Taste gedrückt wird (*kbhit()* gibt 0 zurück, solange keine Taste gedrückt wurde).
7. In jedem Schleifendurchlauf wird die Bildschirmfläche gelöscht (*cleardevice()*) und die Funktion *drawRotatingRectangle()* wird aufgerufen, um ein rotierendes Rechteck anzuzeigen.
8. Der Winkel wird mit 0,1 erhöht, um eine Rotation zu erzeugen.
9. Es wird eine Pause von 160 Millisekunden eingelegt, bevor der nächste Schl
10. Nach dem Verlassen der Schleife wird das Grafiksystem mit *closegraph()* geschlossen und die Funktion kehrt zurück 0 als Rückgabewert zurück.

Es ist üblich, die main()

-Funktion in C-Programmen am Ende des Codes zu platzieren. Das hat technische
main()
 aufgerufen werden, zuvor deklariert werden müssen. Wenn man sie am Anfang des
main() platzieren, wodurch der Code lesbarer und verständlicher wird.

Man kann Funktionen und Variablen in separate Header-Dateien auslagern und sie

**Date Created**
7. April 2023
**Author**
sven