# Programmentwurf Programmieren in C
# Dokumentation

Gruppe STG-TINF18C
Februar 2019
Dozent: Herbert Osterrieder

Nora Baitinger
Matrikelnummer: 2125287

Studiengang: Informatik
Duale Hochschule Baden-Württemberg Stuttgart

# Inhalt

# 1. Liste der Dateien

Das Programm besteht aus der „2125287 Programmentwurf finddup.c“ Datei. Deren Aufbau nun genau erklärt wird:

## 1.1. Kopf der Datei

### 1.1.1. Header-Dateien und C-Bibliotheken

Folgender Header-Dateien werden im Programm verwendet:

| **Header-Datei** | **Funktion** |
|---|---|
| #include <stdio.h> | stdio steht für Standard Inupt Output. Dieser Header wird für Bildschirmein- und ausgaben benötigt. |
| #include <stdlib.h> | stdlib ist die Standardbibliothek. Bestimmte Variablen und Makros sind hier definiert. |
| #include <string.h> | string wird für Stringfunktionen wie strcpy und strcmp verwendet. |
| #include <dirent.h> | dirent wird für die Verwendung von Verzeichnissen und Dateien benötigt. In diesem Programm wird eine Struktur vom Typ dirent benutzt, um alle Dateien in einem Verzeichnis auszulesen. |
| #include <sys\stat.h> | sys/stat wird ebenfalls beim Arbeiten mit Dateien verwendet. Die Struktur stat ermöglicht das Auslesen der Dateiattribute, wie das Änderungsdatum und die Dateigröße. |
| #include <time.h> | time ist ein Header für Zeit und Datum. Im Programm wird der Header für das Auslesen des Änderungsdatums benötigt. |

### 1.1.2. Makros

Mit #define werden Konstanten erstellt. Diese werden im Programm verwendet, um feste Arraygrößen zu verwenden.

### 1.1.3. Globale Strukturen

Im Programm werden zwei Strukturen benötigt. Diese sind global deklariert, um in mehreren Funktionen darauf zuzugreifen.

Die Struktur datei wird als neuer Datentyp definiert. Das heißt im gesamten Programm können Variablen vom Typ datei gespeichert werden. Dieser fasst die benötigten Variablen für eine Datei zusammen.
Zudem können Zeiger auf die Datei-Struktur erstellt werden. Im Programm werden zwei Listen vom Typ datei erstellt. In diesen Listen werden jeweils alle Dateien aus einem Verzeichnis gespeichert.

Die Struktur paare wird verwendet, um ein Array (duplikate[]) aus wahrscheinlich gleichen Dateien zu generieren. In diesem Array werden wahrscheinlich gleiche Dateien als Paare gespeichert und können auf dem Bildschirm ausgegeben und in einer Zieldatei abgespeichert werden.

### 1.1.4. Funktionsprototypen

Damit die Main-Funktion beim Aufruf der Funktionen über deren Existenz aufgeklärt ist, gibt es die Funktionsprototypen. Der Rückgabewert, der Funktionsname und die

Übergabeparameter mit ihrem Typ lassen sich aus dem Funktionsprototyp herauslesen.

Auflistung der Funktionsprototypen meines Programms:

- `void benutzereingabe()`
- `void defaultVorschlag()`
- `int dateiExtTxt(char *)`
- `void zieldateiOk(char *)`
- `void attributeAuslesen()`
- `void vergleich(struct paare *, datei *, datei *)`
- `void zielDateiSchreiben(struct paare *)`
- `void ausgabe(struct paare *)`

### 1.1.5. Globale Variablen

Globale Variablen sind innerhalb der ganzen Datei aufrufbar. Das heißt alle Funktionen können auf ihren Inhalt zugreifen, ohne dass sie als Übergabeparameter übergeben werden müssen. In diesem Programm gibt es fünf globale Variablen, da sie für nahezu alle Funktionen von Bedeutung sind.

Auflistung der globalen Variablen meines Programms:

- `char verzeichnisEins[GROESSE]`
  Diese Variable speichert das erste vom Benutzer eingegebene oder als erster Parameter übergebene Verzeichnis.
- `char verzeichnisZwei[GROESSE]`
  Diese Variable speichert das zweite vom Benutzer eingegebene oder als zweiter Parameter übergebene Verzeichnis.
- `char zielDatei[GROESSE]`
  Diese Variable speichert den Pfad sowie den vom Benutzer eingegebene Dateiname der Zieldatei. Alternativ kann der Pfad und der Dateiname der Zieldatei auch also dritter Parameter an das Programm übergeben werden. Die Zieldatei ist eine Textdatei, die das Ergebnis des Vergleichs in Form eines Textes speichert.
- `char unterverzeichnisse[GROESSE]`
  Diese Variable speichert den Wunsch des Benutzers, ob Unterverzeichnisse in den Vergleich einbezogen werden soll.
  **WICHTIG**: Im Programm ist nur die Variante berücksichtigt, dass der Benutzer „**Nein**“ eingibt. Das beduetet, es können **keine** Unterverzeichnisse miteinbezogen werden.
- `int dup_index`
  Diese Variable speichert die Anzahl der gefundenen wahrscheinlich doppelten Dateien. Diese gehört zur Struktur „paare“ und zum Array „duplikate“, welches die wahrscheinlich doppelten Dateien als Paar speichert.

## 1.2. Hauptprogramm

Das Hauptprogramm wird vom Compiler zuerst aufgerufen und ausgeführt.
Am Anfang des Hauptprogramms stehen die Variablendeklarationen. Anschließend gibt es eine Ausgabe der Kopfdaten jeder Bildschirmausgabe. (Gleiches Vorgehen bei allen Funktionen.)

Das Hauptprogramm bekommt vom Benutzer eventuell Parameter übergeben, die die Suchverzeichnisse, einen Dateipfad und -namen für die Zieldatei und den Wunsch auf Unterverzeichnisse enthält. Das Hauptprogramm prüft, ob diese Parameter angegeben wurden. Wenn dies nicht der Fall ist, wird die Funktion „Benutzereingabe“ aufgerufen. Ansonsten prüft das Hauptprogramm die Sinnhaftigkeit der einzelnen Parameter und weicht gegebenenfalls ebenfalls auf die Benutzereingabe aus. Eine Besonderheit ist die falsche Eingabe der Zieldatei. Das Programm macht einen Vorschlag, den der Benutzer übernehmen kann oder er gibt per Tastatureingabe einen anderen Pfad und Dateinamen an. Dafür wird die Funktion „DefaultVorschlag“ aufgerufen.
Das Hauptprogramm bestimmt ebenfalls den weiteren Fortgang des Programms, indem es die Funktionen „AttributAuslesen“, „ZielDateiSchreiben“ und „Ausgabe“ aufruft.
Der Rückgabewert der Main-Funktion ist „0“.

## 1.3. Funktionen

### 1.3.1. benutzereingabe

Die Funktion „void benutzereingabe()“ dient dazu, die erforderlichen Informationen für den Vergleich vom Benutzer abzufragen. Dabei werden die Eingaben solange wiederholt, bis der Benutzer sinnvolle angaben macht. Das Programm fragt nach zwei Verzeichnispfaden, einem Pfad mit einem Dateinamen, sowie dem Wunsch Unterverzeichnisse einzubeziehen.
Bei der Eingabe des Pfades und des Namens der Zieldatei wird außerdem die Funktion zieldateiOk() aufgerufen, die prüft, ob es sich um eine Datei vom Typ txt handelt.
Die Funktion erhält keine Übergabeparameter und sie hat keinen Rückgabetyp.

### 1.3.2. defaultVorschlag

Die Funktion „void defaultVorschlag()“ wird aufgerufen, um einen Vorschlag zu generieren, wohin und unter welchem Namen die Zieldatei abgespeichert werden soll. Die Funktion verwendet das erste vom Benutzer angegebene Verzeichnis und erstellt dort eine „Ergebnis.txt“-Datei. Der Benutzer kann diesem Vorschlag zustimmen oder per Tastatur einen eigenen Vorschlag eingeben.
Die Funktion erhält keine Übergabeparameter und sie hat keinen Rückgabetyp.

### 1.3.3. dateiExtTxt

Die Funktion „int dateiExtTxt(char *eingabeFileName)“ überprüft, ob es der eingegebene Zieldateiname die Endung txt besitzt. Dafür erhält die Funktion den Namen der Zieldatei als Parameter übergeben. Der Rückgabewert vom Typ Integer kann die Werte „0“ und „1“ annehmen. Diese sagen aus, ob die Endung txt existiert oder nicht.

### 1.3.4. zieldateiOk

Die Funktion „void zieldateiOk(char *eingabeFileName)“ überprüft, ob es sich bei der eingegebenen Zieldatei um eine Datei handelt. Dies wird durch die vorhandene oder eben nicht vorhandene Endung ermittelt. Dabei verwendet „zieldateiOk“ die Funktion „dateiExtTxt“.
Die Funktion hat keinen Rückgabewert.

### 1.3.5. attributeAuslesen

Die Funktion „void attributeAuslesen()“ erhält keine Übergabeparameter und sie hat keinen Rückgabetyp. „

Die Aufgabe der Funktion ist es, zwei Listen anzulegen. Es gibt für jedes vom Benutzer eingegebene Verzeichnis eine eigene Liste. Die Listen sind vom Typ „datei“, da sie alle Dateien enthalten, die im Verzeichnis abgespeichert sind. Außerdem ließt die Funktion für alle Dateien den Dateinamen, die Dateigröße, das Änderungsdatum aus und ermittelt den Dateityp aller Dateien.
Zum Schluss ruft die Funktion dreimal die Funktion „vergleich(struct paare *dup, datei *listeEins, datei *listeZwei)“ auf. Dabei unterscheiden sich die Übergabeparameter bei jedem Aufruf. Zuerst wird zweimal die Startadresse des ersten Verzeichnisses angegeben. Das heißt es soll ein Vergleich nach wahrscheinlich doppelten Dateien innerhalb des ersten Verzeichnisses erfolgen. Anschließen gibt es den Aufruf für den Vergleich innerhalb des zweiten Verzeichnisses und zu Letzt den Funktionsaufruf für den Vergleich beider Verzeichnisse miteinander.

### 1.3.6. vergleich

Die Funktion „void vergleich(struct paare *dup, datei *listeEins, datei *listeZwei)“ hat drei Übergabeparameter und keinen Rückgabewert. Der Parameter „dup“ ist ein Array von der Struktur „paare“. Es soll die zu ermittelnden Duplikate speichern. Die Zeiger vom Typ „datei“ enthalten die Startadressen der zu vergleichenden Listen, die Dateien enthalten. (Es werden auch gleiche Listen miteinander verglichen. Dann haben beide Zeiger die gleichen Startadressen.)
Die Überprüfung auf wahrscheinlich gleiche Dateien erfolgt durch Vergleiche aller Dateien aus der Liste 1 mit allen Dateien aus der Liste 2. Dabei wird überprüft, ob der Dateityp und die Dateigröße identisch sind. Ist das der Fall, dann geht das Programm davon aus, dass die Dateien wahrscheinlich gleich sind.
Wichtig ist dabei, dass beim Vergleich innerhalb des gleichen Verzeichnisses, die Dateien nicht mit sich selbst verglichen werden. Außerdem muss vermieden werden, dass das Programm ein Duplikat (ein Paar aus zwei wahrscheinlich gleichen Dateien) zweimal auflistet. Das kann in Form einer „Kreuzausgabe“ geschehen, wird aber durch das Programm verhindert. „Kreuzausgabe“ bedeutet, dass die Reihenfolge des gefunden Duplikats vertauscht wird (einmal Datei1 mit Datei 2 und als nächstes Duplikat Datei2 mit Datei 1, was das gleiche Duplikat darstellt).
Zudem müssen die „.“ und „..“ Dateien herausgefiltert werden, die diese Referenzen auf „Working-Directory“ und „Parent-Directory“ darstellen.

### 1.3.7. zielDateiSchreiben

Die Funktion „void zielDateiSchreiben(sturct paare *dup)“ erstellt die Zieldatei. Sie speichert die Ergebnisse des Vergleichs in eine Textdatei. Der Benutzer hat Namen und Pfad dafür übergeben oder eingegeben. Die Funktion bekommt das Array „dup“ übergeben, das die Duplikate enthält.

### 1.3.8. ausgabe

Die Funktion „void ausgabe(struct paare *)“ ist vom Aufbau gleich zur Funktion „void zielDateiSchreiben(sturct paare *dup)“. Außerdem erhält auch die Struktur „paare“ als Übergabeparameter, da diese die wahrscheinlich gleichen Dateien als Paare enthält und sie auf dem Bildschirm ausgegeben werden können. Damit erhält der Benutzer zusätzlich zu den Ergebnissen in der Zieldatei, eine Ausgabe der Ergebnisse auf dem Bildschirm.

# 2. Bedienungsanleitung

## 2.1. Eingabemöglichkeit 1: Übergabeparameter an das Programm

Das Programm muss per Eingabeaufforderung gestartet werden und die vier Parameter können dafür direkt dahinter eingegeben werden. Zwischen den einzelnen Übergabeparameter ist ein Leerzeichen.

Eingabe der Parameter:

1. Verzeichnis 1: Beispiel: „D:\Verzeichnis1“
2. Verzeichnis 2: Beispiel: „D:\Verzeichnis2“
3. Pfad und Name der Zieldatei: Beispiel: „D:\Ergebnis.txt“
4. Ist es gewünscht die Unterverzeichnisse zu berücksichtigen?: Beispiel: „Nein“
   **WICHTIG**: Im Programm ist nur die Variante berücksichtigt, dass der Benutzer „**Nein**“ eingibt. Das beduetet, es können **keine** Unterverzeichnisse miteinbezogen werden.

**Beispiel für den Start des Programms über die Eingabeaufforderung**

```
C:\Users\nora\Desktop\2125287>"2125287 Programmierprojekt finddup" "D:\Verzeichnis1" "D:\Verzeichnis2" "D:\Ergebnis.txt" "Nein"
```

### 2.1.1. Korrekte Eingabe aller Übergabeparameter

Nach dem das Programm mit Übergabeparameter gestartet (siehe 2.1) wurde, erscheint direkt die Bildschirmausgabe des Ergebnisses und eine Zieldatei wurde im angegebenen Verzeichnis erstellt.

**Beispiel**:

**Bildschirmausgabe**

```
Finddup - Doppelte Dateien finden
Matrikelnummer: 2125287


Folgende wahrscheinlich doppelte Dateien wurden gefunden:

Verzeichnis 1: D:\Verzeichnis1
Verzeichnis 2: D:\Verzeichnis2

Dateiname          Datum        Typ    Größe    | Dateiname          Datum        Typ    Größe
-----------------------------------------------+-----------------------------------------------
2TEST.txt          10 Feb 2019  .txt   38       | Textdokument22.txt 10 Feb 2019  .txt   38
PDF111.pdf         06 Feb 2019  .pdf   192869   | 222PDF.pdf         06 Feb 2019  .pdf   192869
-----------------------------------------------+-----------------------------------------------
Insgesamt wurden 2 Übereinstimmungen gefunden!
```

**Zieldatei**

```
Ergebnis - Editor
Datei  Bearbeiten  Format  Ansicht  ?
Finddup - Doppelte Dateien finden
Matrikelnummer: 2125287


Folgende wahrscheinlich doppelte Dateien wurden gefunden:
Verzeichnis 1: D:\Verzeichnis1
Verzeichnis 2: D:\Verzeichnis2

Dateiname          Datum        Typ    Groesse  | Dateiname          Datum        Typ    Groesse
-----------------------------------------------+-----------------------------------------------
2TEST.txt          10 Feb 2019  .txt   38       | Textdokument22.txt 10 Feb 2019  .txt   38
PDF111.pdf         06 Feb 2019  .pdf   192869   | 222PDF.pdf         06 Feb 2019  .pdf   192869
-----------------------------------------------+-----------------------------------------------
Insgesamt wurden 2 Uebereinstimmungen gefunden!
```

### 2.1.2. Falsche Eingabe einer oder beider Verzeichnisse und/oder des Wunsches nach Unterverzeichnissen

Der Benutzer wird nun gebeten die benötigten Informationen über die Tastatur einzugeben. Dafür erscheint folgende Aufforderung:

```
Finddup - Doppelte Dateien finden
Matrikelnummer: 2125287


Im folgenden werden Sie aufgefordert, die erforderlichen Informationen für den Vergleich, einzugeben.
Es werden zwei auf Ihrem PC vorhandene Verzeichnisse mit vollständigen Pfadangaben,
sowie ein Verzeichnis mit einem Dateinamen abgefragt.

Bitte geben Sie den Pfad des ersten Verzeichnisses ein (zum Beispiel: C:\Verzeichnis1): D:\Verzeichnis1
Bitte geben Sie den Pfad des zweiten Verzeichnisses ein (zum Beispiel: C:\Verzeichnis2): D:\Ver
Das Verzeichnis konnte nicht geöffnet werden.
Bitte geben Sie erneut einen Pfad für das zweite Verzeichnis ein (zum Beispiel: C:\Verzeichnis2): D:\Verzeichnis2
Bitte geben Sie das Verzeichnis, sowie den Dateinamen der Zieldatei ein,
die das Ergebnis des Vergleichs speichert (zum Beispiel: C:\Ergebnis.txt): D:\Ergebnis.txt
Wollen Sie die Unterverzeichnisse Ihrer Suchverzeichnisse für den Vergleich berücksichtigen?
Geben Sie 'Ja' oder 'Nein' ein: Nein
```

Die Benutzer müssen solange die Eingaben wiederholen bis er eine sinnvolle Eingabe macht.

Beispiel für sinnvolle Eingaben sind aus dem Screenshot ersichtlich.

Daraufhin erscheint wieder das Ausgabefenster des Ergebnisses und die Zieldatei wird am eingegebenen Pfad und mit angegebenem Dateiname erstellt. Diese sehen wie im Punkt 2.1.1 aus.

### 2.1.3. Falsche Eingabe des Pfades und des Namens der Zieldatei

Das Programm macht einen Vorschlag für den Pfad und den Namen der Zieldatei. Der Benutzer kann darauf reagieren.

**Beispiel:**

```
Finddup - Doppelte Dateien finden
Matrikelnummer: 2125287

Die Angabe zur Zieldatei ist nicht korrekt. Soll die Datei unter D:\Verzeichnis1\Ergebnis.txt erstellt werden?
Bitte geben Sie 'Ja' oder 'Nein' ein!
```

Der Benutzer kann mit „Ja“ antworten und der Vorschlag wird übernommen. Dann wird das Ausgabefenster erscheinen und die Zieldatei wird am vorgeschlagenen Pfad und Dateiname erstellt. Diese sehen wie im Punkt 2.1.1 aus.

Wenn der Benutzer mit „Nein“ antwortet folgt, die Eingabe für Pfad und Dateiname der Zieldatei.

**Beispiel**:

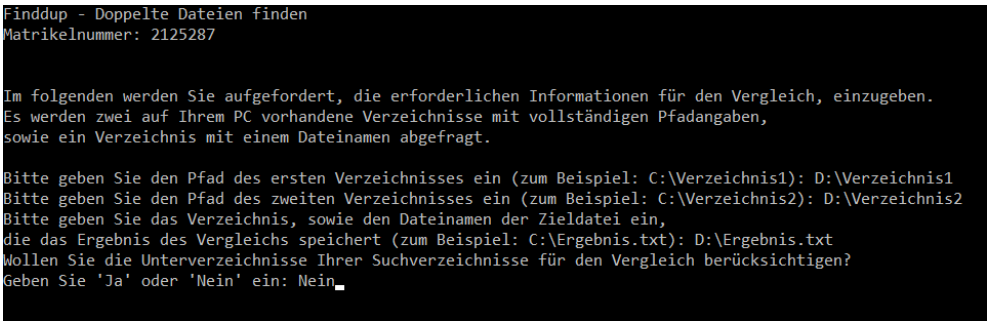
```
Finddup - Doppelte Dateien finden
Matrikelnummer: 2125287

Die Angabe zur Zieldatei ist nicht korrekt. Soll die Datei unter D:\Verzeichnis1\Ergebnis.txt erstellt werden?
Bitte geben Sie 'Ja' oder 'Nein' ein! Nein
Bitte geben Sie das Verzeichnis, sowie den Dateinamen der Zieldatei ein,
die das Ergebnis des Vergleichs speichert (zum Beispiel: C:\Ergebnis.txt): D:\Ergebnis.txt
```

## 2.2. Eingabemöglichkeit 2: Benutzereingabe

### 2.2.1. Korrekte Angaben

Wenn keine Angaben für die Übergabeparameter gemacht werden oder diese fehlerhaft sind, kann der Benutzer die Informationen auch über die Tastatur eingeben.




Für die einzelnen Werte müssen folgende Angaben gemacht werden:

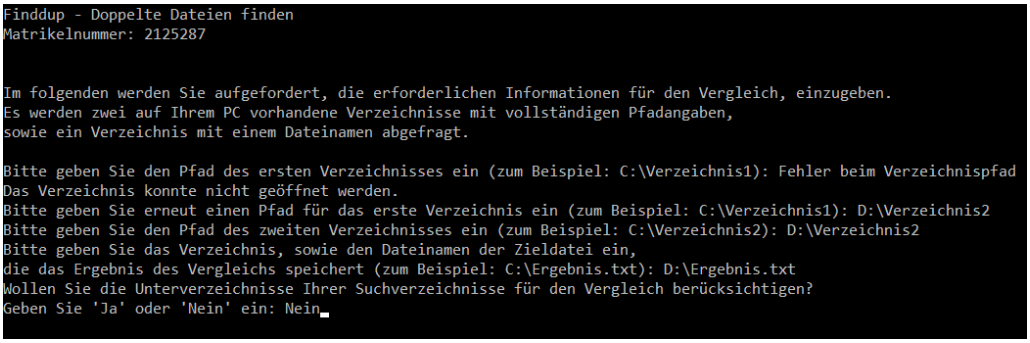
1. Verzeichnis 1: Beispiel: D:\Verzeichnis1
2. Verzeichnis 2: Beispiel: D:\Verzeichnis2
3. Pfad und Name der Zieldatei: Beispiel: D:\Ergebnis.txt
4. Ist es gewünscht die Unterverzeichnisse zu berücksichtigen?: Beispiel: Nein
   **WICHTIG**: Im Programm ist nur die Variante berücksichtigt, dass der Benutzer **Nein** eingibt. Das beduetet, es können **keine** Unterverzeichnisse miteinbezogen werden.

Da alle Angaben korrekt sind, gibt es die Bildschirmausgabe des Ergebnisses und die Zieldatei wird angelegt. Beispiele sind unter 2.1.1 aufgeführt.

### 2.2.2. Fehlerhafte Eingabe

Wenn der Benutzer dabei Fehler macht, muss er die Eingabe wiederholen bis er eine sinnvolle Angabe macht.

**Beispiel**:

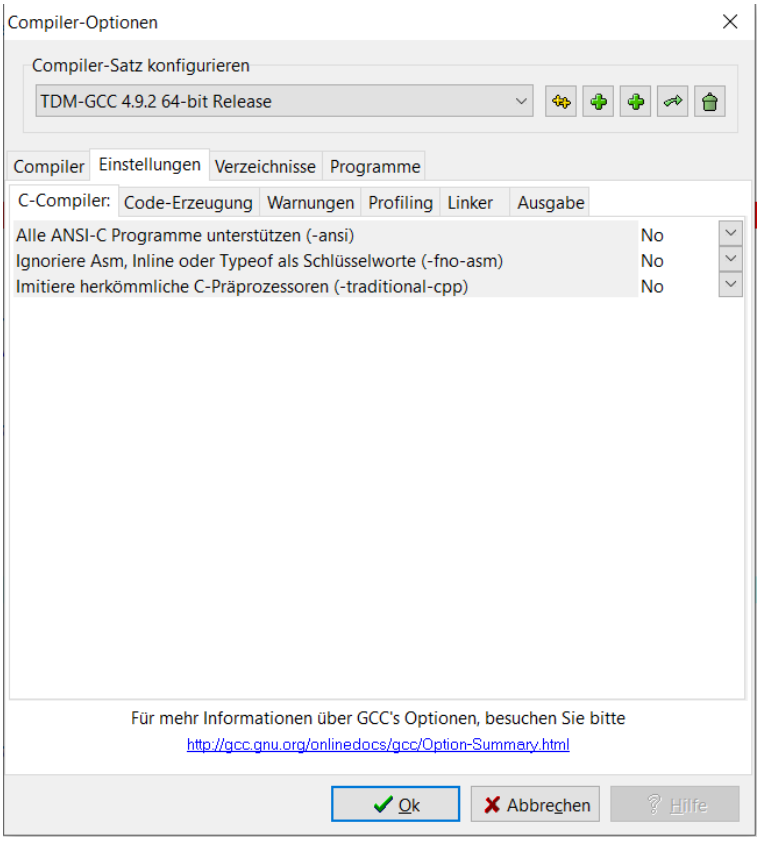


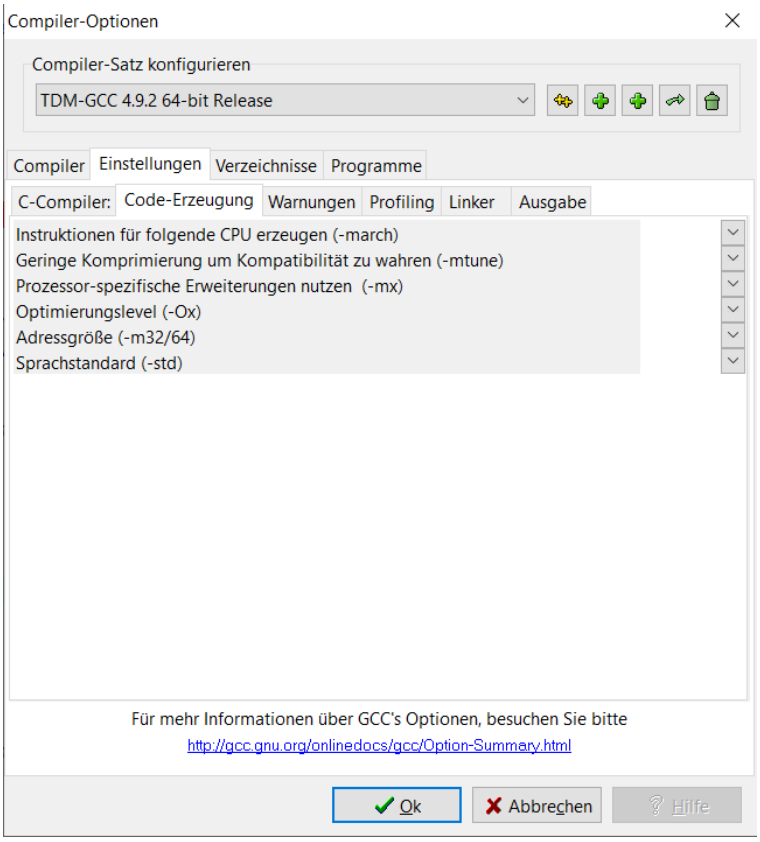

Eine Besonderheit ist der Default-Vorschlag beim Namen und Pfad der Zieldatei. Das wird im Punkt 2.1.3 erläutert und ein Beispiel gegeben.

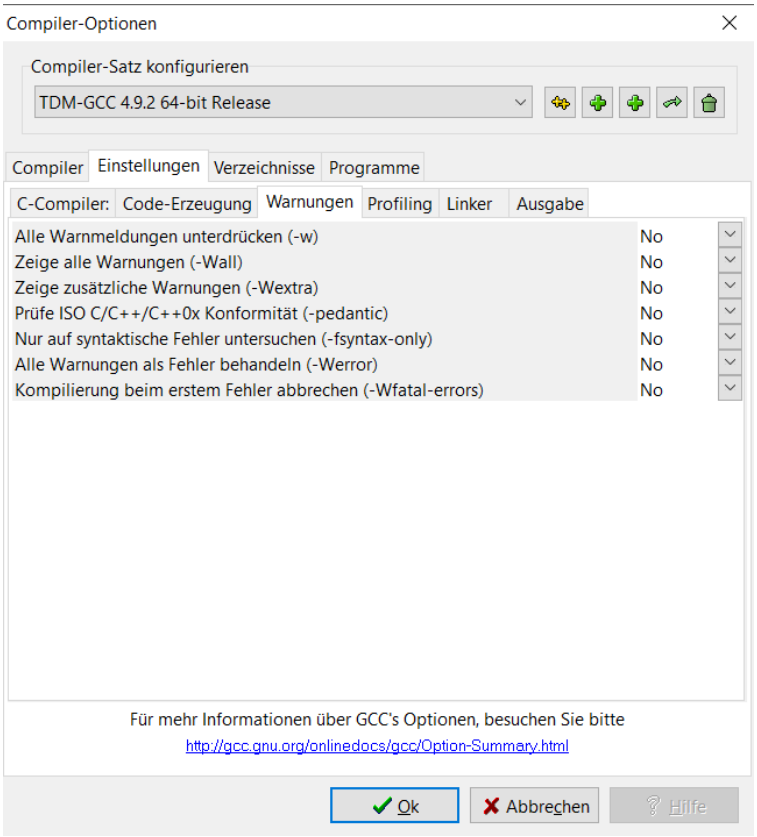
# 3. Kompiliereinstellungen

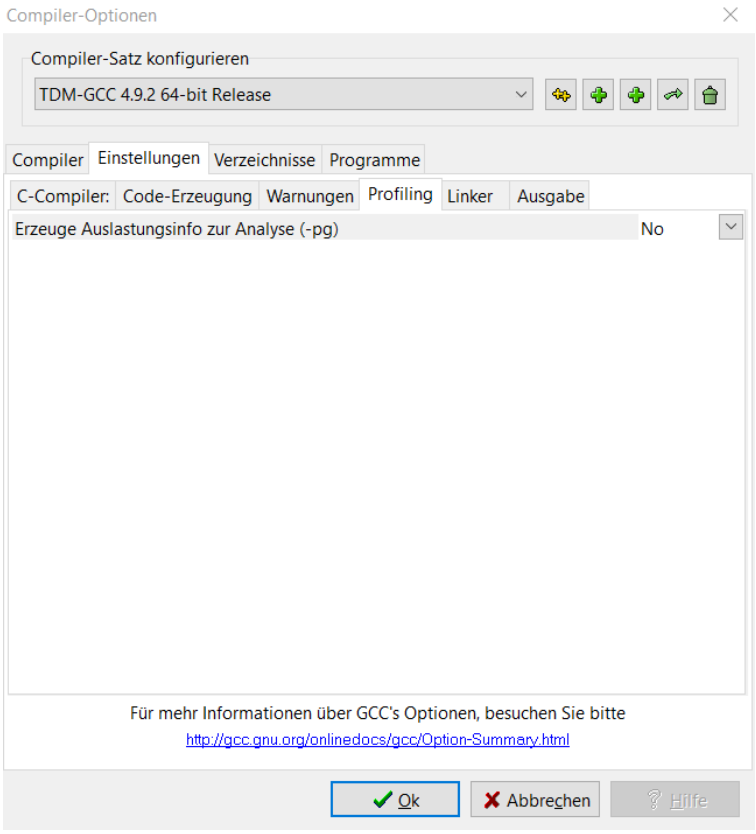
Für die Programmierung des Programms wurde das Programm DevC++ und dessen Compiler verwendet.

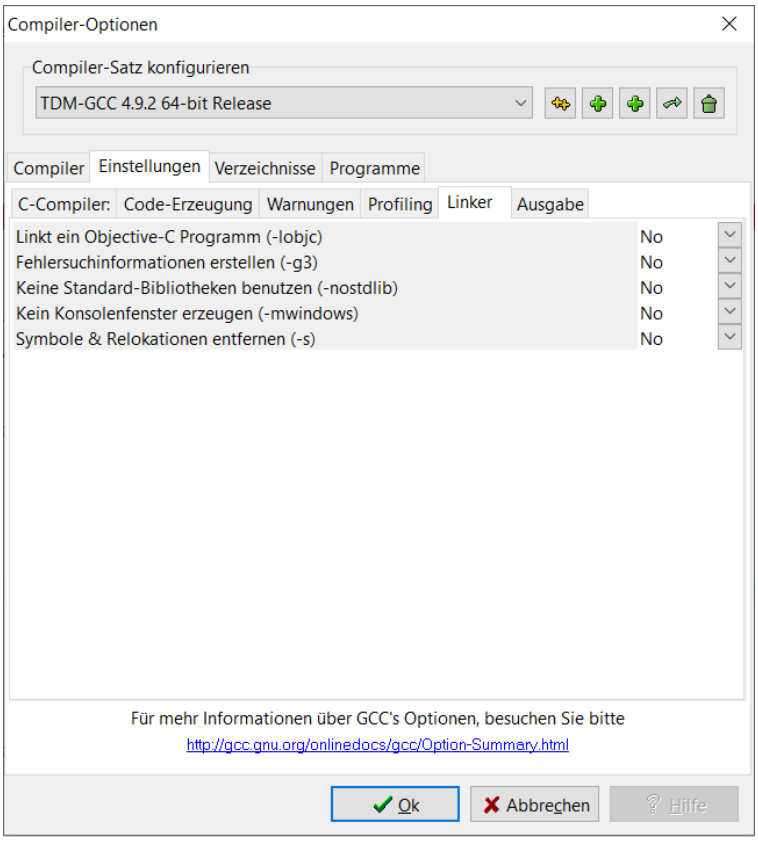
Die Einstellungen des Compilers sind aus folgenden Screenshots ersichtlich:

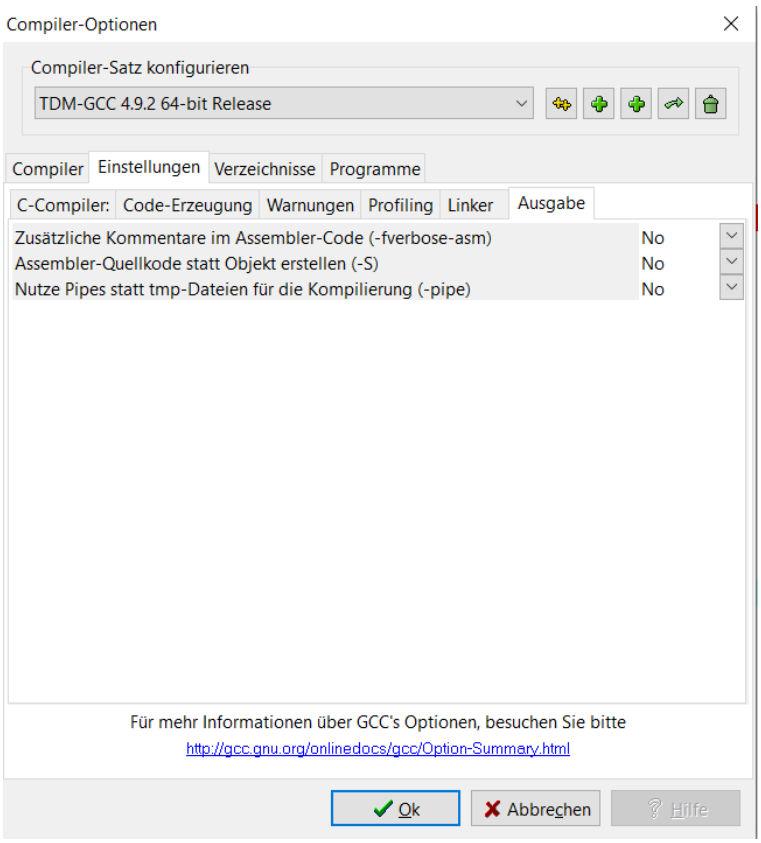



Compiler-Optionen
Compiler-Satz konfigurieren
TDM-GCC 4.9.2 64-bit Release
Compiler
Einstellungen
Verzeichnisse
Programme
C-Compiler:
Code-Erzeugung
Warnungen
Profiling
Linker
Ausgabe
Alle Warnmeldungen unterdrücken (-w) No
Zeige alle Warnungen (-Wall) No
Zeige zusätzliche Warnungen (-Wextra) No
Prüfe ISO C/C++/C++0x Konformität (-pedantic) No
Nur auf syntaktische Fehler untersuchen (-fsyntax-only) No
Alle Warnungen als Fehler behandeln (-Werror) No
Kompilierung beim erstem Fehler abbrechen (-Wfatal-errors) No
Für mehr Informationen über GCC's Optionen, besuchen Sie bitte
http://gcc.gnu.org/onlinedocs/gcc/Option-Summary.html
Ok
Abbrechen
Hilfe


Compiler-Optionen
Compiler-Satz konfigurieren
TDM-GCC 4.9.2 64-bit Release
Compiler
Einstellungen
Verzeichnisse
Programme
C-Compiler:
Code-Erzeugung
Warnungen
Profiling
Linker
Ausgabe
Erzeuge Auslastungsinfo zur Analyse (-pg) No
Für mehr Informationen über GCC's Optionen, besuchen Sie bitte
http://gcc.gnu.org/onlinedocs/gcc/Option-Summary.html
Ok
Abbrechen
Hilfe


Compiler-Optionen
Compiler-Satz konfigurieren
TDM-GCC 4.9.2 64-bit Release
Compiler
Einstellungen
Verzeichnisse
Programme
C-Compiler:
Code-Erzeugung
Warnungen
Profiling
Linker
Ausgabe
Linkt ein Objective-C Programm (-lobjc) No
Fehlersuchinformationen erstellen (-g3) No
Keine Standard-Bibliotheken benutzen (-nostdlib) No
Kein Konsolenfenster erzeugen (-mwindows) No
Symbole & Relokationen entfernen (-s) No
Für mehr Informationen über GCC's Optionen, besuchen Sie bitte
http://gcc.gnu.org/onlinedocs/gcc/Option-Summary.html
Ok
Abbrechen
Hilfe


Compiler-Optionen
Compiler-Satz konfigurieren
TDM-GCC 4.9.2 64-bit Release
Compiler
Einstellungen
Verzeichnisse
Programme
C-Compiler:
Code-Erzeugung
Warnungen
Profiling
Linker
Ausgabe
Zusätzliche Kommentare im Assembler-Code (-fverbose-asm) No
Assembler-Quellkode statt Objekt erstellen (-S) No
Nutze Pipes statt tmp-Dateien für die Kompilierung (-pipe) No
Für mehr Informationen über GCC's Optionen, besuchen Sie bitte
http://gcc.gnu.org/onlinedocs/gcc/Option-Summary.html
Ok
Abbrechen
Hilfe