BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
DRESDEN
UNIVERSITY OF COOPERATIVE EDUCATION

Imperative Programmierung

Übungsaufgaben

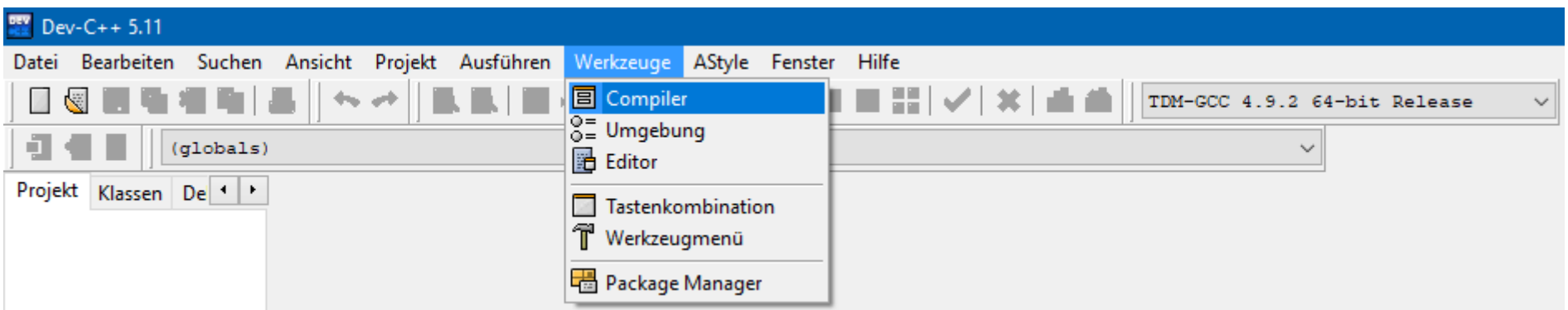
Prof. Dr.-Ing. Tenshi Hara
tenshi.hara@ba-sachsen.de

Einrichtung der Entwicklungs-umgebung

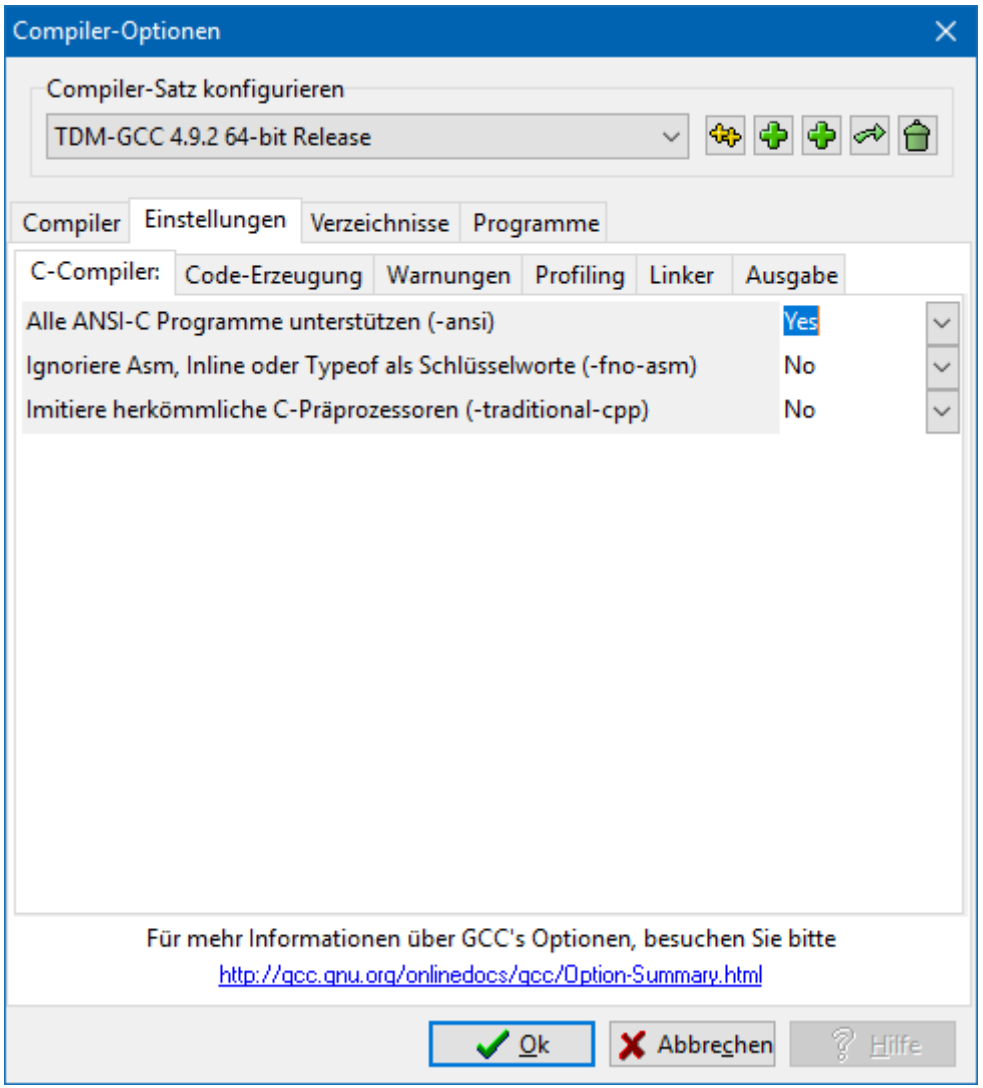
AUFGABE

Richten Sie Ihre Entwicklungsumgebung ein, damit Sie später C-Programme kompilieren und testen können.

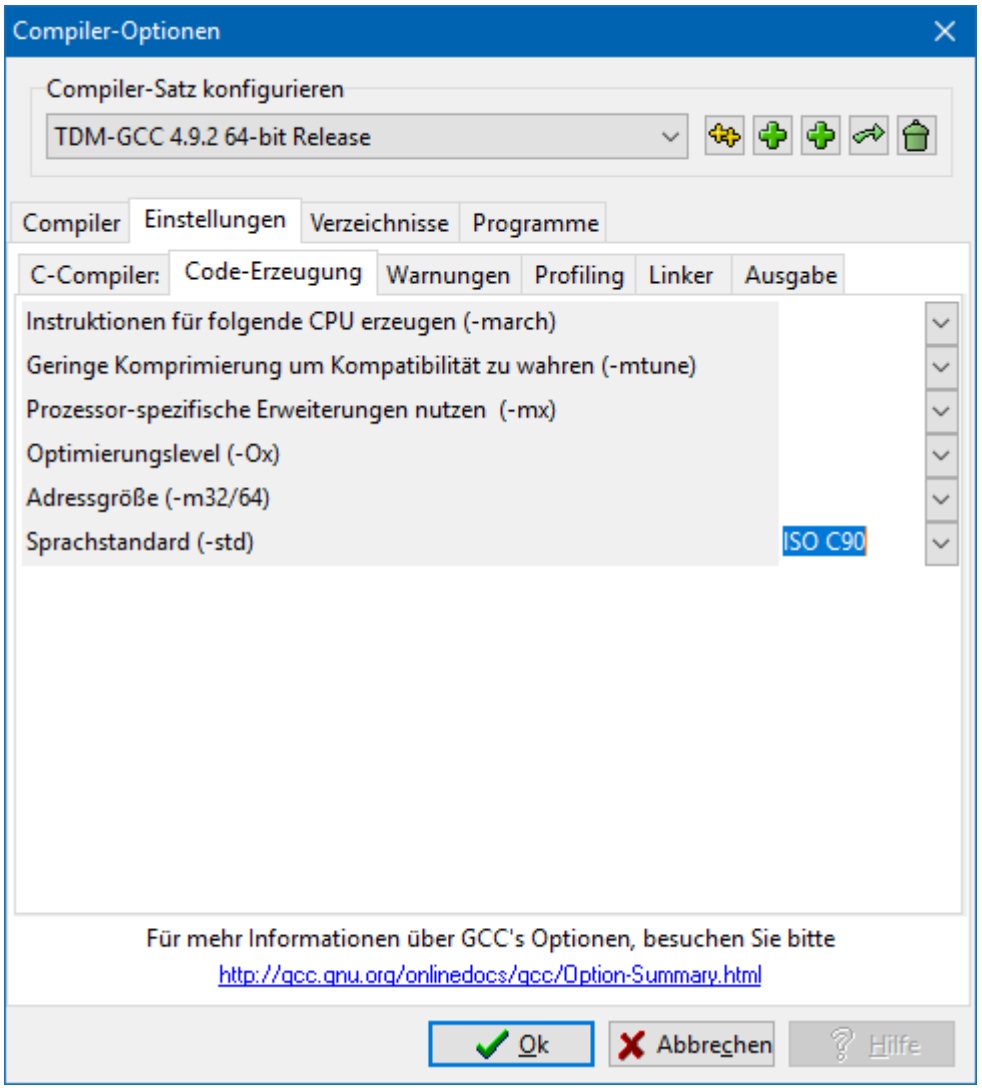
Dev-C++: auf ANSI-C einstellen (1/4)



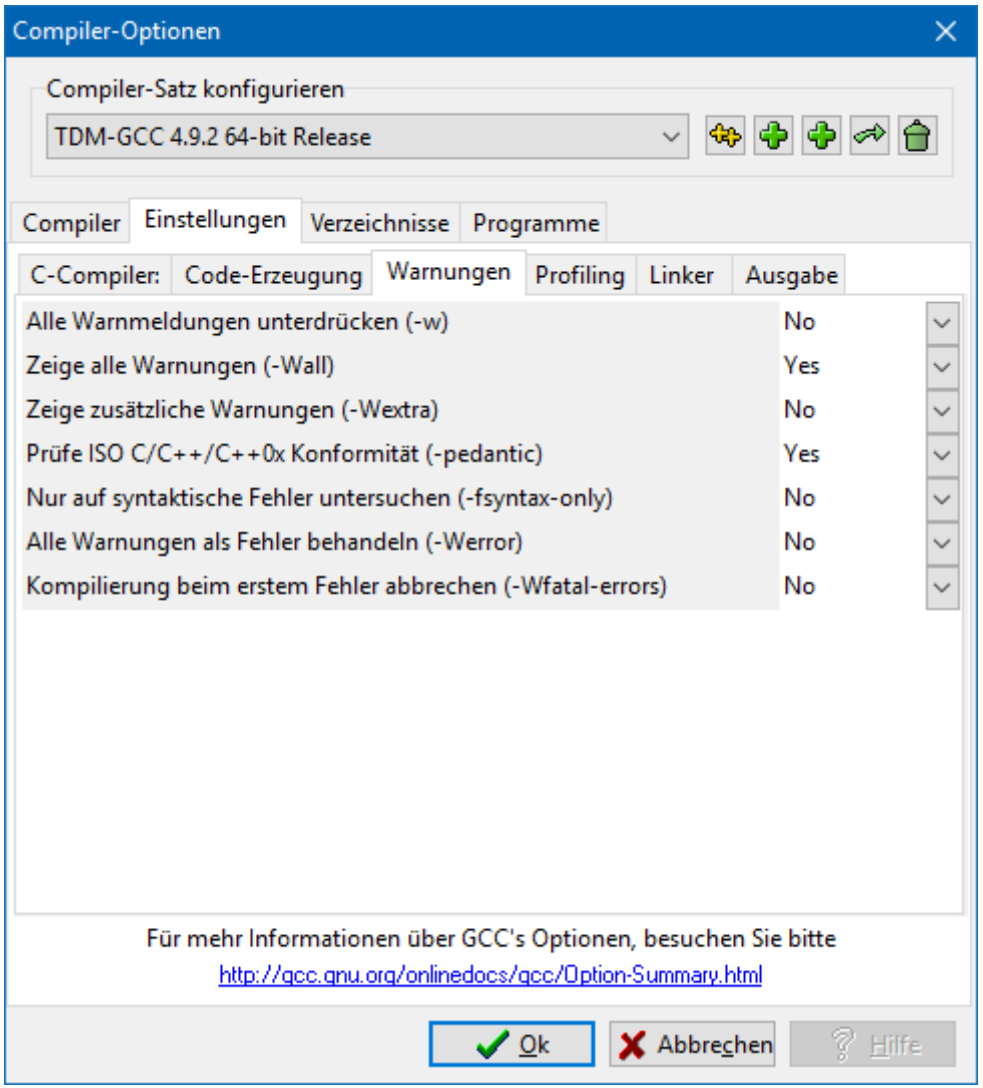
DEV-C++: AUF ANSI-C EINSTELLEN (2/4)



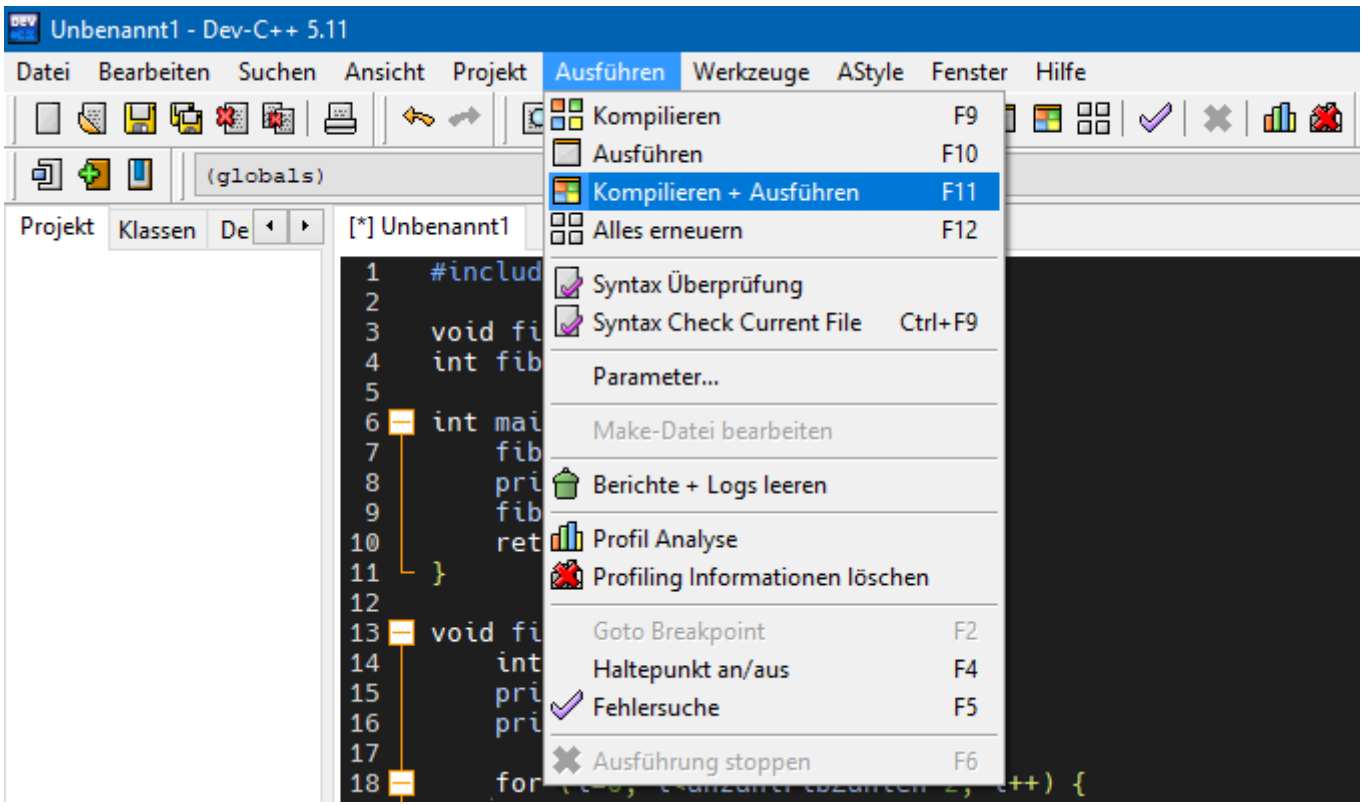
DEV-C++: AUF ANSI-C EINSTELLEN (3/4)



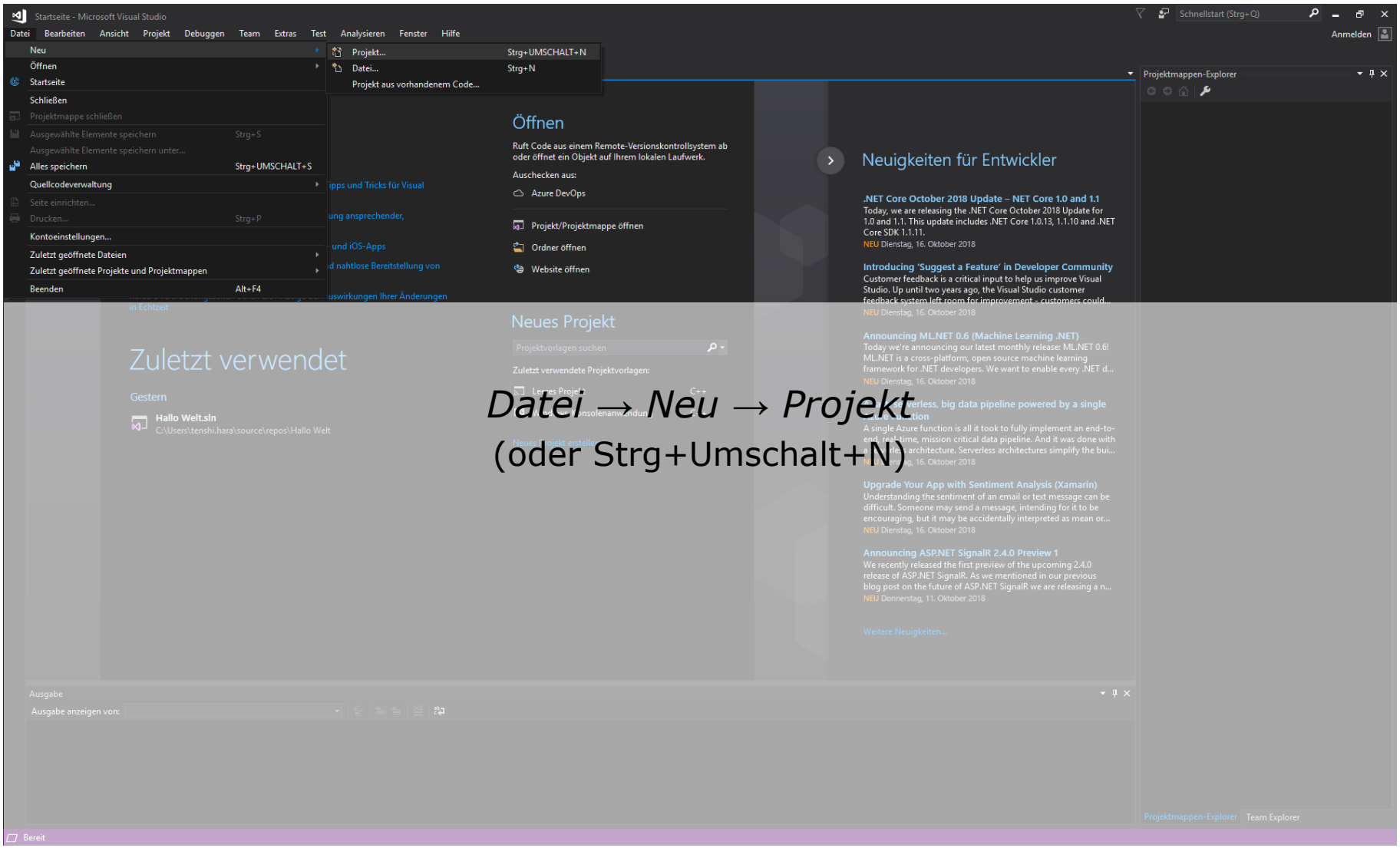
Dev-C++: auf ANSI-C einstellen (4/4)



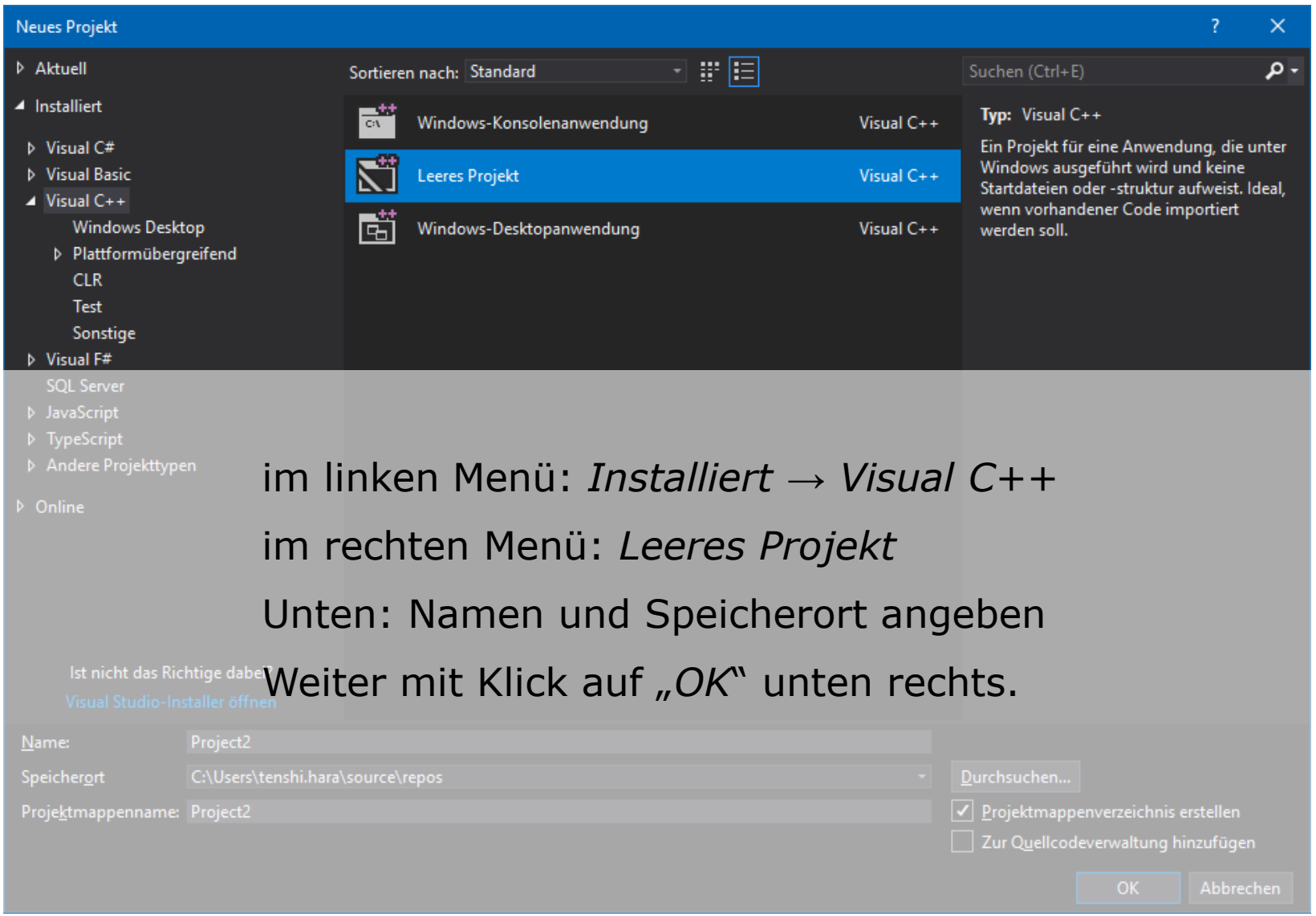
Dev-C++: Programm kompilieren und ausführen



Visual Studio: neues C-Projekt anlegen (1/4)



Visual Studio: neues C-Projekt anlegen (2/4)



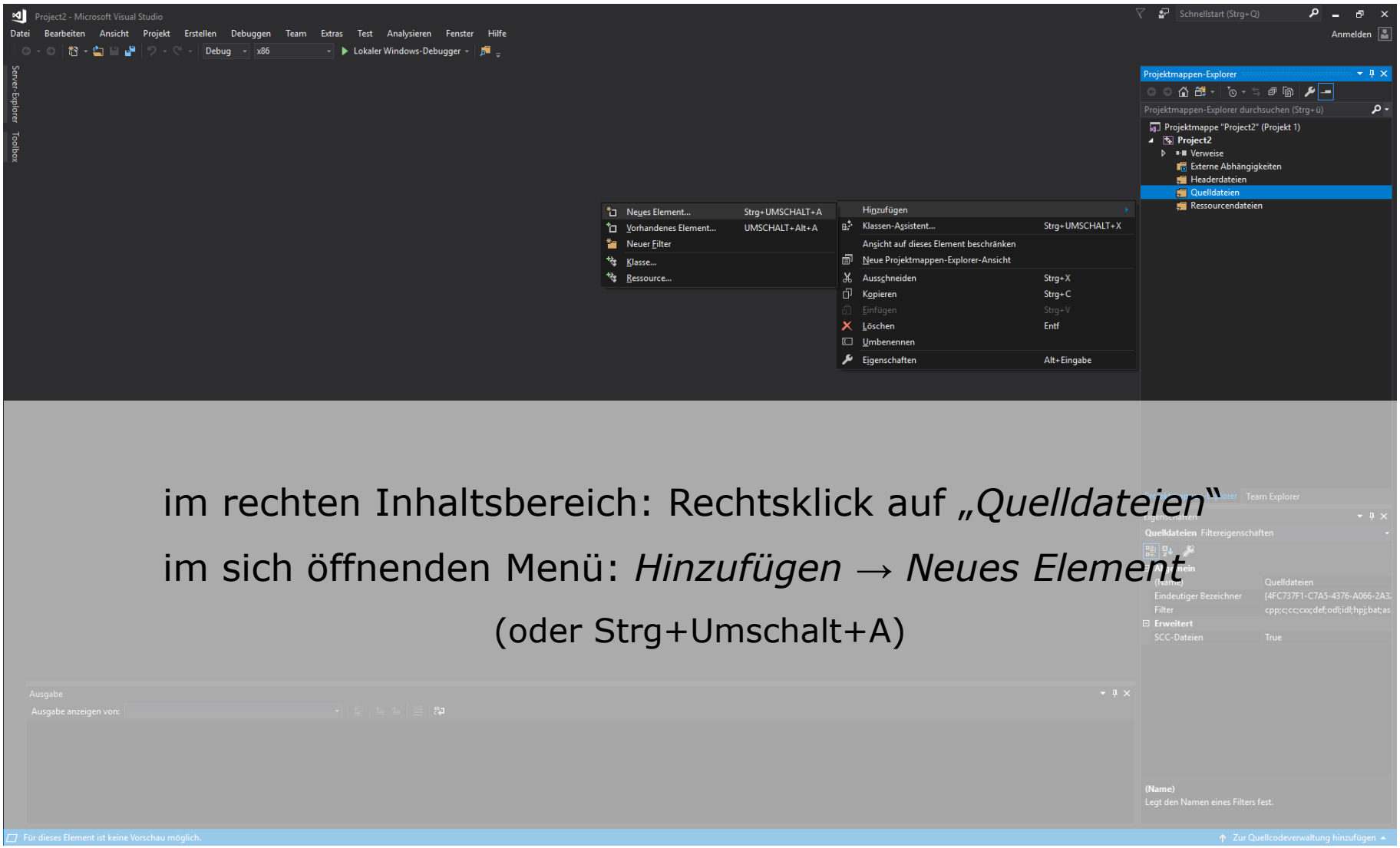
im linken Menü: *Installiert → Visual C++*

im rechten Menü: *Leeres Projekt*

Unten: Namen und Speicherort angeben

Weiter mit Klick auf „*OK*“ unten rechts.

Visual Studio: neues C-Projekt anlegen (3/4)

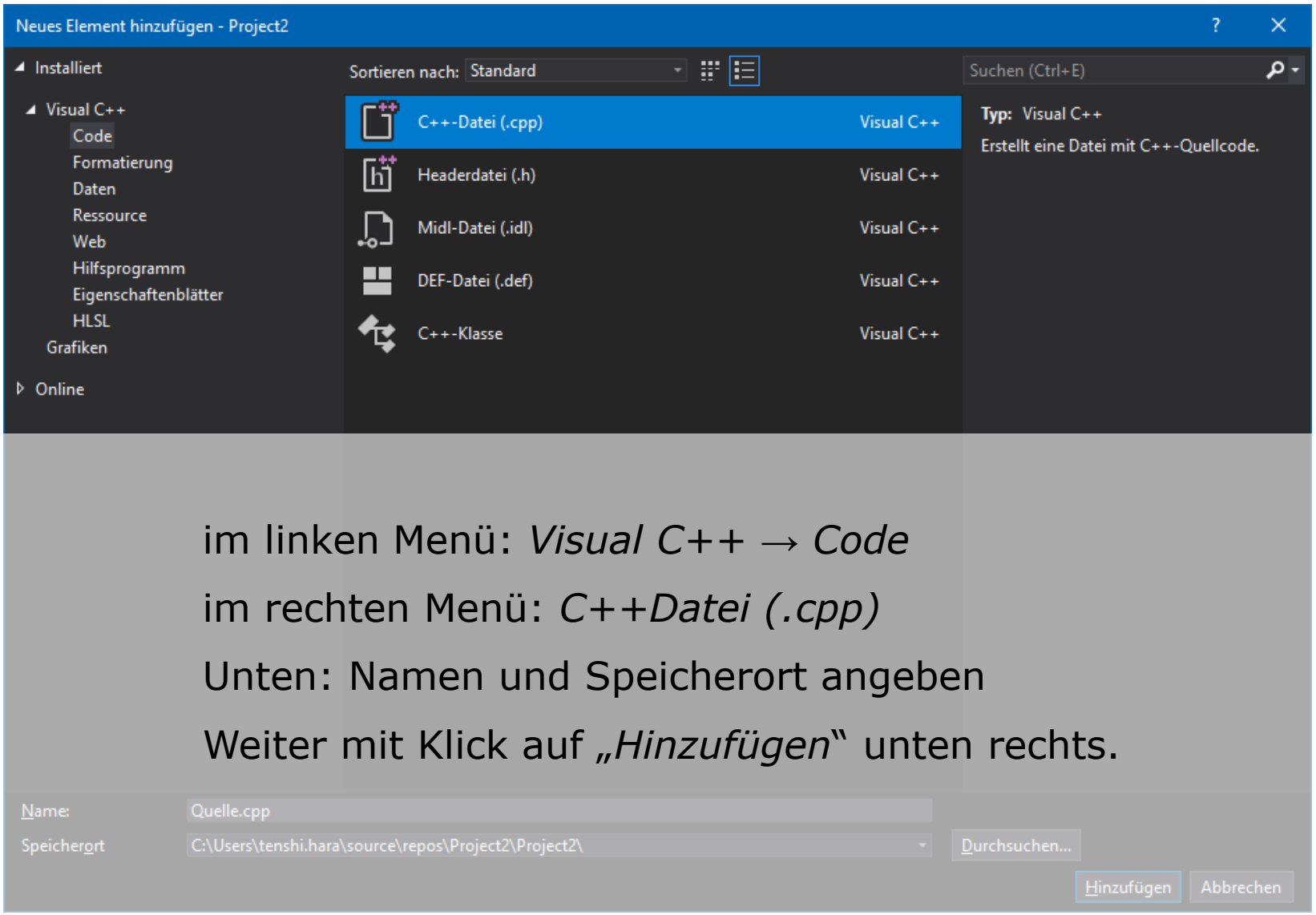


im rechten Inhaltsbereich: Rechtsklick auf „*Quelldateien*“

im sich öffnenden Menü: *Hinzufügen* → *Neues Element*

(oder Strg+Umschalt+A)

Visual Studio: neues C-Projekt anlegen (4/4)



im linken Menü: *Visual C++ → Code*

im rechten Menü: *C++Datei (.cpp)*

Unten: Namen und Speicherort angeben

Weiter mit Klick auf „*Hinzufügen*“ unten rechts.

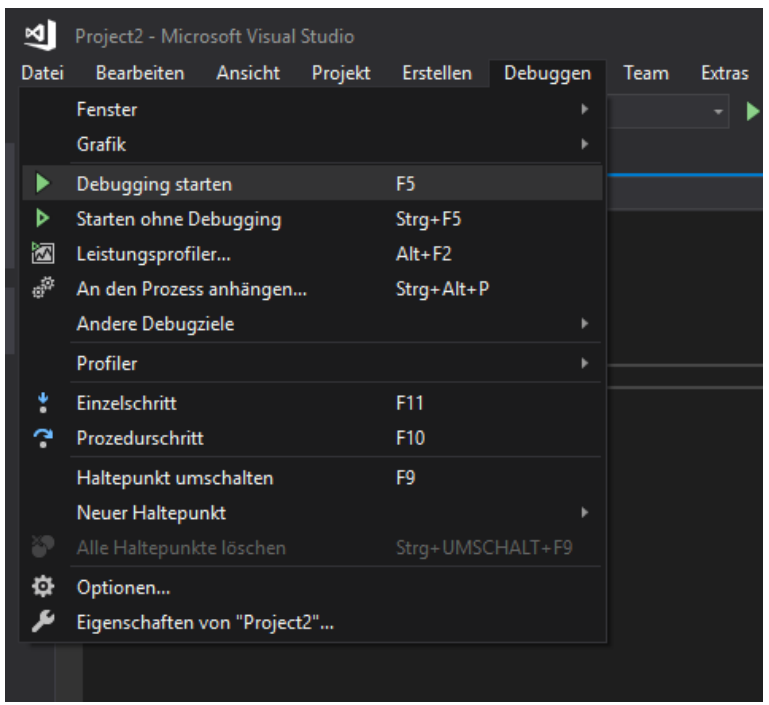
Visual Studio: Programm kompilieren und ausführen

Mit einem Doppelklick auf Ihre neue Datei im rechten Inhaltsbereich (bei den *Quelldateien*) öffnen Sie einen Bereich zur Eingabe Ihres Quellkodes.

Ihren Quellkode können Sie über

Debuggen → *Starten ohne Debugging*
(oder Strg+F5)

kompilieren und ausführen.



Notepad++ mit MinGW portable (1/2)

- FAQ bei MinGW beachten
- eine neue Batch-Datei anlegen (z.B. `mingw.bat`) mit dem folgenden Inhalt:

```
@echo off
set npppath=%~1
set nppexec=%~2
if exist "%npppath%\%nppexec%.exe" del "%npppath%\%nppexec%.exe"
rem zum USB-Stick mit MinGW wechseln, z.B. mit dem Kommando D:
rem zum Verzeichnis mit MinGW wechseln, z.B. mit cd MinGW\bin
gcc -o "%npppath%\%nppexec%.exe" "%npppath%\%nppexec%.c"
if exist "%npppath%\%nppexec%.exe" (
    %comspec% /c "%npppath%\%nppexec%.exe"
) else (
    echo Es ist ein Fehler bei der Kompilierung aufgetreten.
)
pause
exit
```

Notepad++ mit MinGW portable (2/2)

- in Notepad++ einen neuen Ausführen-Shortcut anlegen
- z.B. für die Tastenkombination Ctrl+Shift+F5
- dem Shortcut den folgenden Kode zuweisen:

```
D:\mingw &quot;$(CURRENT_DIRECTORY)&quot;
&quot;$(NAME_PART)&quot;
```

- Den Kode müssen Sie entsprechend dem Speicherort (falls nicht `D:`) und dem Namen Ihrer Batch-Datei (falls nicht `mingw.bat`) anpassen!
- Wenn Sie ein Programm kompilieren und ausführen wollen, müssen Sie im Notepad++-Fenster nur noch Strg+Umschalt+F5 betätigen.

GNU Compiler Collection (gcc)

- Sie können gcc unter Linux verwenden indem Sie Ihre gespeicherte C-Kode-Dateien dem gcc übergeben und dabei auf ANSI-C prüfen:

```
gcc -Wall -ansi -pedantic -Wmissing-prototypes -Wstrict-prototypes -Wold-style-definition kodedatei -o zielbinary
```

- Ihnen steht in den Laboren eine virtuelle Maschine mit einem Ubuntu-Linux zur Verfügung. Doppelklicken Sie dazu einfach auf Ihrem Desktop auf die Verknüpfung „TensorFlow“.

Vorgehensweise für alle zu erstellenden Programme

Diagramme und Pseudokode

Zu jedem Programm gehört mindestens ein Entwurf!

Für alle Übungsaufgaben gelten daher die folgenden Schritte:

1. Erstellen Sie auf Papier vorher einen PAP, ein Struktogramm oder Pseudokode für Ihren Algorithmus.
2. Prüfen Sie auf dem Papier, ob Ihr so spezifizierter Algorithmus lauffähig sein könnte!
   → Korrigieren Sie nötigenfalls bis Sie zufrieden sind.
3. Implementieren Sie das Programm.
4. Prüfen Sie das Programm auf Fehler.
   → Korrigieren Sie nötigenfalls bis Sie keine Fehler mehr finden.
5. Prüfen Sie, ob Ihr Programm wirklich das tut, was es tun soll.
   (Und ggf. ob es NUR das tut, was es tun soll.)
6. Vergleichen Sie das fertige Programm mit Ihrem Entwurf.
   → Korrigieren Sie nötigenfalls das Diagramm / den Pseudokode.

Hallo Welt!

Aufgabe

Implementieren Sie ein Programm zur Ausgabe des Textes „Hallo Welt!“ in der Konsole.

Vergessen Sie nicht, dass Sie eine `main`-Funktion benötigen, mit der Ihr C-Programm startet!

simple Zahlen

AUFGABE

Implementieren Sie ein Programm zur Ausführung einiger beispielhafter arithmetischer Operationen (Addition, Subtraktion, …).

Deklarieren Sie dazu ein paar Variablen eines geeigneten Formats! – Beachten Sie, dass die Division gegebenenfalls nicht zu einem ganzzahligen Ergebnis führt.

Die Ergebnisse der Operationen sollen in der Konsole in geeigneter Form ausgegeben werden. → Googlen Sie ggf. nach der printf()-Funktion!

Funktionen

AUFGABE

Implementieren Sie ein Programm, das neben der main()-Funktion zwei weitere Funktionen enthält. Die erste Funktion soll die von Ihnen bereits implementierte „Hallo Welt!“-Ausgabe erledigen, die zweite soll die von Ihnen bereits implementierten arithmetischen Operationen mitsamt Ausgabe erledigen.

AUFGABE

Ändern Sie Ihr Programm mit Funktionen nun in der Art ab, dass nur noch lokale Variablen verwendet werden.

Ergänzen Sie Ihr Programm um Kommentare!

AUFGABE

Entfernen Sie nun den „Hallo Welt!“-Teil aus Ihrem Programm.

Ändern Sie die Funktion für die arithmetischen Operationen in der Form ab, dass der Funktion Parameter übergeben werden können und, dass die Rückgabe vom Typ Integer ist.

Der Aufruf der Funktionen für arithmetische Operationen soll nun aus der main()-Methode mit Parameterübergabe erfolgen. Beachten Sie dabei die klare Trennung zwischen globalen und lokalen Variablen!

Kontrollstrukturen

AUFGABE

Implementieren Sie ein neues Programm zur Veranschaulichung von bedingten Verzweigungen und von Mehrfachverzweigungen.

Damit es halbwegs interessant ist und der Programmablauf nicht von Anfang an klar ist, nutzen Sie eine Zufallszahl als Prüfkriterium!

- Sie benötigen die Bibliothek `stdlib.h`.
- Sie benötigen die Bibliothek `time.h`.
- Ihr Hauptprogramm benötigt einmalig vor dem Erzeugen der ersten Zufallszahl die Anweisung `srand(time(0));`.
- Schränken Sie die Urmenge der Zufallszahl durch geeignete Operationen ein, bspw. `rand() % 100`.

Aufgabe

Ergänzen Sie das Programm mit den Verzweigungen um einfache Zählschleifen, kopfgesteuerten Schleifen und fußgesteuerten Schleifen.

Nutzen Sie geeignete Kodewiederverwendung durch Funktionen!

Umsetzungsbeispiel:

- 10 Zufallszahlen generieren, dann
- solange Zufallszahlen generieren, bis eine besondere dabei ist, dann
- solange insgesamt weniger als 20 Zufallszahlen generiert wurden, weitere Zufallszahlen generieren.

Iteration und Rekursion

AUFGABE

Implementieren Sie eine Funktion `int fakul_i(int)` zur iterativen Berechnung der Fakultät einer natürlichen Zahl oder von 0.

Implementieren Sie anschließend eine Funktion `int fakul_r(int)` die selbiges rekursiv erledigt.

Gehen Sie in beiden Fällen davon aus, dass die Zahl korrekt an die Funktionen übergeben wird.

AUFGABE

Implementieren Sie eine Funktion `int fib_i(int)` zur iterativen Berechnung der n-ten Fibonacci-Zahl.

Implementieren Sie anschließend eine Funktion `int fib_r(int)` die selbiges rekursiv erledigt.

Gehen Sie in beiden Fällen davon aus, dass n korrekt als natürliche Zahl eingegeben wird.

größter gemeinsamer Teiler

AUFGABE

Implementieren Sie eine Funktion `int ggT_k (int, int)` zur Berechnung des größten gemeinsamen Teilers (ggT) zweier natürlicher Zahlen inklusive der 0 auf Basis des klassischen Euklid'schen Algorithmus (Subtraktionsverfahren).

Geben Sie die Ergebnisse der Berechnungsschritte über die Konsole aus!

Der ggT soll im Hauptprogramm (`main`-Funktion) ausgegeben werden.

AUFGABE

Implementieren Sie eine Funktion `int ggT_m(int, int)` zur Berechnung des ggT zweier natürlicher Zahlen auf Basis des modernen Euklid'schen Algorithmus (Moduloverfahren).

Geben Sie die Ergebnisse der Berechnungsschritte über die Konsole aus!

Der ggT soll im Hauptprogramm (`main`-Funktion) ausgegeben werden.

Aufgabe

Da der Datentyp `int` alle ganzen Zahlen zulässt, also auch negative Zahlen und die `0`, ergänzen Sie Ihre Funktionen um eine Prüfung, ob tatsächliche natürliche Zahlen eingegeben wurden. Falls nicht sollen die Funktionen `-1` als Fehlerwert zurückgeben.

Anschließend implementieren Sie die Funktion für das Moduloverfahren als `int ggT_r(int, int, int)` unter Nutzung von Rekursion neu. Dabei soll ein Parameter `ausgabe` die Ausgabe der einzelnen Berechnungsschritte deaktivieren können (`ausgabe == 0`).

AUFGABE

Implementieren Sie eine Funktion `int kgV(int, int)`, die das kleinste gemeinsame Vielfache zweier natürlicher Zahlen an das Hauptprogramm zurückgibt.

Eine Ausgabe von Berechnungsschritten soll nicht erfolgen.

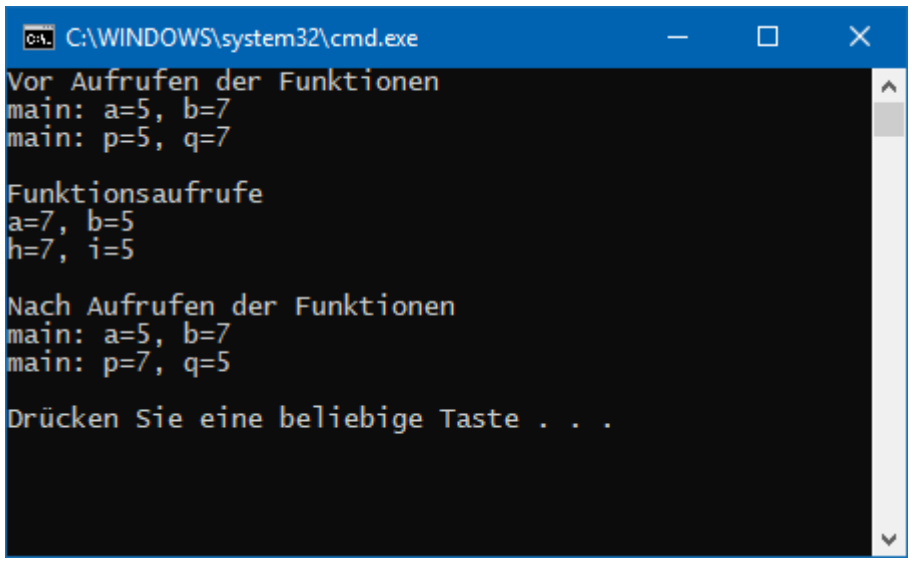
Call-by-Value und Call-by-Reference

AUFGABE

Implementieren Sie ein C-Programm zum Demonstrieren des Unterschiedes zwischen Call-by-Value und Call-by-Reference indem Sie die Inhalte zweier Variablen in entsprechenden Funktionen vertauschen.

- Implementieren Sie ein Hauptprogramm mit der Ausgabe der Variablenbelegungen vor und nach den Funktionsaufrufen.
- Implementieren Sie eine Funktion `void tauschValue(int, int)`, welche das Vertauschen im Call-by-Value-Sinne erledigt.
- Implementieren Sie eine Funktion `void tauschReference(int, int)`, welche das Vertauschen im Call-by-Reference-Sinne erledigt.
- Vergleichen Sie die Ausgaben im Hauptprogramm vor und nach den Funktionsaufrufen!

Call-by-Value und Call-by-Reference



Felder

AUFGABE

Implementieren Sie ein C-Programm zum Umgang mit Feldern!

1. Definieren Sie zwei Felder unterschiedlichen Typs (`feld1`, `feld2`).

2. Zeigen Sie, dass der Feldtyp wie bei einer „normalen“ Variable tatsächlich von Bedeutung ist.
   (bspw. können Sie Elemente der Felder miteinander vergleichen)

Texteingaben und naive Strings

AUFGABE

Implementieren Sie ein C-Programm zum naiven Umgang mit Strings!

Verwenden Sie **nicht** die String-Bibliothek oder fertige String-Funktionen!

1. Es sollen zwei Zeichenketten (`text1`, `text2`) von der Kommandozeile aus eingelesen werden.
2. Die erste Zeichenkette soll vollständig in eine neue Variable (`zieltext`) kopiert werden.
3. Die zweite Zeichenkette soll vollständig rechtsseitig an den Inhalt der neuen Variable konkateniert werden.
   (Ziel der Aufgabe in Pseudokode: zieltext := text1 ⊙ text2)

Zeichenketten

AUFGABEN

- Zeichenketten werden in üblicherweise mit Standardfunktionen bearbeitet (z.B. `strcpy`, `strlen`, `strcat`, `strcmp`, …)
- dazu muss der Header `#include <string.h>` eingefügt werden
- Standardfunktionen verarbeiten Enden von Zeichenkette korrekt (d.h. korrekter Umgang mit binärer Null (`\0`) am Ende von Strings)

Aufgaben:

1. Auf den folgenden drei Folien sehen Sie naive Implementierungen von diversen String-Funktionen. Implementieren Sie diese Funktionen als Varianten mit Zeigern!
2. Wie lauten die Funktionsaufrufe bei der Verwendung der Standardbibliothek `string.h`?

Texte kopieren

```
void textKopie(char ziel[], char quelle[]) {
   int i;
   for (i=0; quelle[i]; i++)
      ziel[i] = quelle[i];
   ziel[i] = quelle[i];
}

void textKopie(char ziel[], char quelle[]) {
   int i = 0;
   do {
      ziel[i] = quelle[i];
   } while (ziel[i++]);
}
```

Textlänge bestimmen und Zeichenketten anhängen

```
int textLaenge(char ziel[]) {
   int lang = 0;

   while (ziel[lang]) lang++;

   return lang;
}

void textAnhang(char ziel[], char quelle[]) {
   int i=0, j=0;

   while (ziel[i]) i++;

   do {
      ziel[i] = quelle[j++];
   } while (ziel[i++]);
}
```

Zeichenkettenvergleich

```
/*
 * Rückkehrwert = 0: Zeichenketten sind gleich
 * Rückkehrwert > 0: 1. Zeichenkette ist lexikografisch
 *                   größer als die 2. Zeichenkette
 * Rückkehrwert < 0: umgekehrt
 */
int textVergleich(char text1[], char text2[]) {
   int i=0;

   while (text1[i] == text2[i]) {
      if (text1[i] == '\0')
         return 0;
      i++;
   }

   return (text1[i] - text2[i]);
}
```

Aufgaben

1. Auf den vorherigen drei Folien sahen Sie naive Implementierungen von diversen String-Funktionen. Implementieren Sie diese Funktionen als Varianten mit Zeigern!

2. Wie lauten die Funktionsaufrufe bei der Verwendung der Standardbibliothek `string.h`?

Aufgaben

Aufgaben:

1. Auf den vorherigen drei Folien sahen Sie naive Implementierungen von diversen String-Funktionen. Implementieren Sie diese Funktionen als Varianten mit Zeigern!
2. Wie lauten die Funktionsaufrufe bei der Verwendung der Standardbibliothek `string.h`?

MIT STANDARDBIBLIOTHEK STRING.H

- Textkopie
  - Nutzung von: `char *strcpy(char *dest, const char *src)`
  - Aufruf: `strcpy(ziel, "zu Kopierendes");`
- Textlänge
  - Nutzung von: `size_t strlen(const char *str)`
  - Aufruf: `int len = strlen("Wie lang ist dieser Text?");`
- Text anhängen (Konkatenation)
  - Nutzung von: `char *strcat(char *dest, const char *src)`
  - Aufruf: `strcat("Apfel", "kuchen");`
- Texte vergleichen
  - Nutzung von: `int strcmp(const char *str1, const char *str2)`
  - Aufruf: `int ret = strcmp("sehr kurzer Text", "langer Text");`

MIT STANDARDBIBLIOTHEK STRING.H

weitere Infos

- https://www.tutorialspoint.com/c_standard_library/string_h.htm
- https://www2.hs-fulda.de/~klingebiel/c-stdlib/string.htm
- https://openbook.rheinwerk-verlag.de/c_von_a_bis_z/030_c_anhang_b_020.htm#mj687f8a374bbc05a0aec2b47a307aa4ba

Kodeanalyse

AUFGABE

```
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

bool check(const char *pPurported) {
   int nSum       = 0;
   int nDigits    = strlen(pPurported);
   int nParity    = (nDigits-1) % 2;
   char cDigit[2] = "\0";
   for (int i=nDigits; i>0 ; i--) {
      cDigit[0]  = pPurported[i-1];

      // Die atoi()-Funktion konvertiert einen String in Integer.
      // Der String muss \0-terminierend sein (s.o.).
      int nDigit = atoi(cDigit);

      if (nParity == i % 2)
         nDigit = nDigit * 2;

      nSum += nDigit / 10;
      nSum += nDigit % 10;
   }
   return 0 == nSum % 10;
}
```

Analysieren Sie nebenstehenden Kode. Was macht die check( )-Funktion?

Konsolen-Taschenrechner

AUFGABE

Implementieren Sie mit den Ihnen bisher bekannten Mitteln einen Konsolentaschenrechner für Gleitkommazahlen!

Der Taschenrechner soll haben

- mindestens einen Operand
- maximal zwei Operanden
- die folgenden Operatoren
  - `+` … Addition
  - `–` … Subtraktion
  - `*` … Multiplikation
  - `/` … Division
  - `w` … Quadratwurzel
  - `s` … Sinus
  - `c` … Cosinus
- Fehlerbehandlung (bspw. Division durch 0 oder negativer Radikant)
- Möglichkeit, mehrere Rechnungen nacheinander auszuführen

Deklarationen

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int berechne(double, double, char, double *);
void fehlerBehandlung(int);
```

FEHLERBEHANDLUNG

```c
void fehlerBehandlung(int fehler) {
   if (fehler == 0) return;

   switch(fehler) {
      case 1: printf("Divison durch Null\n");
              break;
      case 2: printf("Wurzel einer negativen Zahl\n");
              break;
      case 3: printf("Unbekannte Rechenoperation\n");
              printf("(erlaubt: +, -, *, /, w, s oder c)\n");
              break;
   }
}
```

simple Studierenden-Datenbank

AUFGABE

Implementieren Sie eine (simple) Studierenden-Datenbank zur Erfassung von Nachname, Vorname und Wohnort (jeweils maximal 16 Buchstaben). Nutzen Sie dazu eine doppelt-verkettete Liste und das Grundgerüst der Datei `simple_db_empty.c`.

Ihr Programm soll mindestens die folgenden Funktionen bieten:

- Datensatz anlegen
- nächster Datensatz
- vorheriger Datensatz
- aktuellen Datensatz anzeigen
- aktuellen Datensatz löschen
- alle Datensätze anzeigen
- alle Datensätze löschen
- Datensätze alphabetisch aufsteigend nach Nachnamen sortieren
- Suche nach dem ersten Treffer in einem Suchkriterium

Doppelt verkettete Liste (DVL)