Einführung

Vorbereitung

Im Praktikum sollten Sie weiterhin CMake verwenden, um die für Ihre Entwicklungsumgebung benötigte Projektmappen zu erzeugen. Im vergangenen Semester haben Sie dies bereits installiert und verwendet. Falls nicht, finden Sie weiter unten kurze Installationsanleitungen.

Ein CMake-Tutorial finden Sie hier[1]. Im wesentlichen benötigen Sie zunächst lediglich aus dem Abschnitt „A Basic Starting Point (Step 1)“ die Einleitung und die Schritte „Specify the C++ Standard“ und „Build and Test“.

Alternativ können Sie selbstverständlich direkt eine IDE Ihrer Wahl verwenden. Machen Sie sich in dem Fall bitte selbst vertraut damit, wie Sie C++-Anwendungen für die Kommandozeile erstellen. Wie auch im vergangenen Semester können Sie selbstverständlich nach wie vor Compiler und Linker aus der Kommandozeile bedienen, um Ihre Programme zu bauen. Beachten Sie bitte, dass Sie nun `g++` (statt `gcc`) bzw. `clang++` (statt `clang`) verwenden, um C++-Programme zu erstellen.

Windows

- Laden Sie die passende Installer-Binary-Distribution herunter[2].
- Installieren Sie CMake.

macOS

- Installieren Sie CMake im Terminal über Homebrew[3] mittels `brew install cmake`.
- Alternative: Laden Sie die macOS Binary-Distribution herunter[2]. Kopieren Sie die enthaltene Applikation ins zentrale Programm-Verzeichnis.

Ubuntu 18.04 LTS

- Installieren Sie CMake im Terminal über den Paketmanager mittels `sudo apt-get install cmake`.

centos7

- Laden Sie die Linux-Distribution als `.tar.gz`-Archiv herunter[2].
- Entpacken Sie das Archiv und kopieren Sie den Inhalt in ein sinnvolles Verzeichnis Ihrer Wahl, z. B. `/opt/cmake/`.
- Fügen Sie den absoluten Pfad des enthaltenen Unterverzeichnis `bin` der Umgebungsvariable `PATH` hinzu.

centos8

- Installieren Sie CMake im Terminal über den Paketmanager mittels `sudo dnf install cmake`.

Einführung in C++

1. **Hello-World**
   (a) Erzeugen Sie mit CMake eine Projektmappe für die von Ihnen verwendete Entwicklungsumgebung. Sie soll lediglich die Datei `main.cpp` enthalten.
   (b) Erstellen Sie in C++ ein Hello-World-Programm.

   Wiederholen Sie die nötigen Schritte mehrfach.

[1] https://cmake.org/cmake/help/latest/guide/tutorial/index.html
[2] `https://cmake.org/download/`
[3] `https://brew.sh`

2. Taschenrechner

Implementieren Sie einen einfachen Taschenrechner.

(a) Rechenanweisungen der Form `[Zahl][Operator][Zahl]` werden für Ganzzahlen sowie die Operatoren `+`, `-`, `*`, `/` von der Tastatur eingelesen.

(b) Die Eingabe wird mit der Enter- bzw. der Return-Taste abgeschlossen.

(c) Abhängig vom Operator wird von Ihrem Programm die passende Berechnung durchgeführt, anschließend wird das Ergebnis der Rechnung am Bildschirm in einer neuen Zeile ausgegeben.

(d) Danach kann eine neue Rechnung eingegeben werden, bis dem Programm z.B. durch Eingabe von Strg+D signalisiert wird, dass der Benutzer keine weiteren Rechenanweisungen eingeben, sondern das Programm beenden möchte.

(e) Um den vorherigen Punkt erfüllen zu können, prüft Ihr Programm nach dem Einlesen einer Rechenanweisung, ob es dem Stream `std::cin` „gutgeht“[a]. Falls nicht, wird das Programm verlassen.

[a] `https://en.cppreference.com/w/cpp/io/basic_ios/good`

Ist alles implementiert, sollte die Eingabe links die Ausgabe rechts erzeugen – bei manueller Eingabe jeweils zeilenweise abwechselnd:

Eingabe:

```
3+4
9*13
1*12
27/9
8*4
```

Ausgabe:

```
7
117
12
3
32
```

Am Ende jeder Zeile steht ein Zeilenwechsel.

In der Kommandozeile ist es mittels *Pipes* möglich, den Inhalt einer Datei quasi als Tastatureingabe an ein anderes Programm weiterzugeben. Die Ausgabe eines Programms kann in eine Datei umgeleitet werden. Unter Linux/macOS sollte folgende Befehlszeile den Inhalt der Datei `input.txt` als Tastatureingabe an `./main` weitergeben. Dessen Ausgabe landet anschließend in der Datei `output.txt`:

```
cat input.txt | ./main > output.txt
```

Unter Windows sollten Sie `type` anstelle von `cat` verwenden können.

Vertiefung

3. Namensräume

(a) Implementieren Sie die folgende Funktion in der Datei `main.cpp`:

```
void tell();
```

Diese Funktion soll `Hello!` am Bildschirm ausgeben.
Rufen Sie diese Funktion aus Ihrer `main`-Funktion auf.
Erstellen und testen Sie Ihr Programm.

(b) Implementieren Sie die folgende Funktion in der Datei `main.cpp`:

```
void tell();
```

Der Name ist tatsächlich identisch zur Funktion vorher. Diese Funktion soll `Goodbye!` am Bildschirm ausgeben.
Rufen Sie diese Funktion aus Ihrer `main`-Funktion nach der vorherigen Funktion auf.
Erstellen Sie Ihr Programm. Welche Fehlermeldung erhalten Sie von welchem an der Erstellung beteiligten Programm?

(c) Verlagern Sie die Funktionen in jeweils einen eigenen Namensraum – nach wie vor in der Datei `main.cpp`. Rufen Sie beide Funktionen entsprechend auf.
Erstellen und testen Sie Ihr Programm.

(d) Erstellen Sie für jede der beiden `tell`-Funktionen ein eigenes Paar aus Header (`.hpp`) und Quelltextdatei (`.cpp`). Legen Sie die Funktionen dort *ohne* Namensräume an. Entfernen Sie die Funktionen aus `main.cpp`.
Binden Sie die entsprechenden Header in `main.cpp` ein. Ändern Sie ggf. die Aufrufe in Ihrer `main`-Funktion.
Erstellen Sie Ihr Programm. Achten Sie darauf, dass alle Header- und Quelltextdateien berücksichtigt werden.
Welche Fehlermeldung erhalten Sie nun von welchem an der Erstellung beteiligten Programm?

(e) Verlagern Sie die `tell`-Funktionen in jeweils einen eigenen Namensraum in den jeweiligen Dateien. Ändern Sie ggf. die Aufrufe in Ihrer `main`-Funktion.
Erstellen und testen Sie Ihr Programm.

4. Referenzen

(a) Implementieren und testen Sie eine `swap`-Funktion einmal mit Zeigern und einmal mit Referenzen entsprechend dem Beispiel in der Vorlesung (Sommersemester 2020).

(b) Gegeben seien zwei Integer-Variablen `a` und `b`. Implementieren Sie eine Funktion `void order`, der diese Variablen übergeben werden können. Direkt nach Aufruf der Funktion soll folgende Bedingung erfüllt sein:

$$a \leq b$$

Implementieren Sie die Funktion einmal mit Zeigern, einmal mit Referenzen.