Inhalt

1 Einführung

Herzlich willkommen zu diesem Kurs und wie schön, dass du eine Programmiersprache lernen möchtest. Ich freue mich sehr darauf, dich ein Stück weit auf diesem Weg begleiten zu dürfen!

Intro

Das Ziel dieses Lernhefts ist einfach: Wenn du alles gelesen und vor allem die Beispiele und Aufgaben bearbeitet hast, dann wirst du eigene Programme schreiben und fremden Quellcode (z.B. auf *GitHub*) verstehen können. Das Besondere dabei ist:

- **Praxisnahe Projekte:** Das Wissen in diesem Kurs wird entlang von Projekten vermittelt. Wenn du ein neues Konzept lernst (z.B. *Variablen* oder *Schleifen*), dann wirst du es unmittelbar danach in der Praxis anwenden und etwas Sinnvolles damit tun.
- **YouTube-Videos:** Die Inhalte werden durch zahlreiche Videos und Programmierbeispiele unterstützt, damit dein Wissen sich schnell verfestigt.
- **Coden im Browser:** Du kannst die Beispiele aus dem Heft in deinem Browser anzeigen, verändern und ausführen. So kannst du direkt mit dem Programmieren loslegen, ohne vorher auf deinem Rechner etwas installieren zu müssen.

Auf den folgenden Seiten lernst du an einer der bekanntesten und leistungsfähigsten Sprache der Welt die wichtigsten Grundlagen der Programmierung kennen. Dazu zählen z.B. *Variablen, Verzweigungen, Schleifen* und *Klassen*. Und das Schöne ist: Wenn du C++ beherrschst, dann fällt es dir leicht, dich schnell in andere Sprachen wie Java oder Python einzuarbeiten.

Wenn du mit mir Kontakt aufnehmen möchtest, oder Fragen hast, dann schreib gerne an `kontakt@andreashaja.com`! Und jetzt viel Spaß bei deinen ersten Schritten in die Welt von C++!

Prof. Dr. Andreas Haja

1.1 Leitfaden für dieses Lernheft

Damit du dich gut zurecht findest, habe ich mir die folgenden Formatierungen für dich überlegt:

```
1 In dieser Umgebung stehen die Code-Beispiele.
2
3 Jede Zeile hat hier eine eindeutige Nummer.
4
5 Und wenn du den QR-Code einscannst, dann kannst du
6 das Beispiel direkt in deinem Browser ausführen.
```

Listing 1.1: Dies ist ein Code-Beispiel

Code

Definition 1.1.1. *Wichtige Informationen sind so wie in diesem Beispiel markiert. Damit fallen dir die wirklich relevanten Dinge sofort ins Auge.*

Übung 1.1.1. *An mehreren Stellen im Lernheft findest du Übungen, mit denen du deinen Lernfortschritt kontrollieren und verfestigen kannst. Diese sind links mit einem Balken markiert und in kursiver Schrift gesetzt.*

Außerdem gibt es hin und wieder Zusatzinformationen, die zwar interessant, aber für deinen Lernerfolg nicht so relevant sind. Diese findest du in kleiner Schrift.

Zusätzlich zu den **online verfügbaren Listings** gibt es an vielen Stellen Videos für dich. Dabei kann es sich um eine Einführung in ein Thema handeln oder ich erkläre dir einen Zusammenhang oder programmiere ein Beispiel. Genau wie bei den Listings musst du dazu nur die entsprechenden QR-Codes einscannen[1]. Diese findest du in der Regel an der Seite.

Am Ende jedes Kapitels findest du einige größere **Aufgaben und Lernkontrollfragen**. Diese kannst du genau wie die Listings in deinem Browser bearbeiten. Das Schöne dabei: Du kannst direkt überprüfen, ob du richtig geantwortet bzw. programmiert hast. Aufgaben und Fragen sind auch über einen QR-Code zugänglich.

Updates

Noch ein letzter Hinweis, bevor es losgeht: Damit du von nachträglichen Erweiterungen des Kurses profitieren kannst, findest du unter dem QR-Code **Updates** eine Liste von Themen, die nach dem Kauf deines Lernhefts noch ergänzt wurden. Schau gelegentlich vorbei, ob es etwas Neues gibt.

1.2 Wie lernst du richtig?

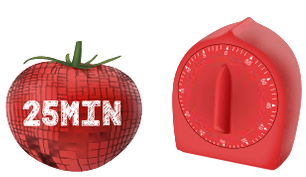
Pomodoro

Das Lernen einer Programmiersprache ist nicht einfach. Das gilt besonders dann, wenn es sich um deine erste Sprache handelt. Damit dir der Einstieg leichter fällt und du nicht irgendwann frustriert aufhörst, möchte ich dir in diesem Abschnitt eine Lernhilfe empfehlen: Die **Pomodoro-Technik**.

Das Prinzip dieser Methode ist denkbar einfach: Phasen konzentrierter Arbeit wechseln sich ab mit regelmäßigen kurzen und längeren Pausen. Damit ist es möglich, produktiver und länger zu arbeiten und dabei greifbare Fortschritte zu erzielen.

Für dich heißt das konkret:

1. Du wählst ein Teilkapitel oder ein Thema aus, mit dem du dich beschäftigen willst.
2. Du stellst dir einen Wecker auf 25 Minuten und machst dich ans Werk. Dabei bitte keine Unterbrechungen zulassen und wirklich fokussieren. Da 25 Minuten ein überschaubarer Zeitraum sind, wird dir das auch gelingen.



3. Wenn der Wecker klingelt, machst du 5 Minuten Pause. Dabei ist es egal, was du machst, aber es sollte dir Spaß bringen und vor allem deinen Kopf entspannen.
4. Wenn die Pause vorbei ist, stellst du dir den Wecker wieder auf 25 Minuten und weiter geht's mit dem Lernen.
5. Spätestens nach dem vierten Durchgang solltest du eine längere Pause von ca. 30 Minuten machen, sonst besteht die Gefahr, dass deine Konzentration zu stark abnimmt.

[1] Falls du dieses Heft in digitaler Form hast, dann kannst du die Codes auch einfach anklicken!

2 Die wichtigsten Programmier-Tools

2.1 Worum geht es in diesem Kapitel?

Eine der großen Hürden am Anfang besteht darin, die richtigen Tools zum Programmieren zu finden und auf dem eigenen Rechner zum laufen zu bekommen. In diesem Modul wird dir daher eine Reihe von unverzichtbaren Tools empfohlen, mit denen du deinen Rechner fit für das Programmieren machen kannst.

Tools zum Programmieren

Neben einem Compiler zum Übersetzen von Quellcode in Programmen brauchst du eine Entwicklungsumgebung, mit der du deinen Code schreiben kannst. Und du solltest die Kommandozeile benutzen können, z.B. um Pakete zu installieren, den Compiler aufzurufen und deine selbstgeschriebenen Programme zu starten.

Außerdem solltest du (idealerweise) von Beginn an deine Projekte unter Versionsverwaltung stellen, damit du nichts verlierst und bei Fehlern schnell wieder auf einen früheren Stand zurückkehren kannst.

Das kannst du nach diesem Kapitel:

- **Die wichtigsten Befehle für die Kommandozeile benutzen.**
- **Linux unter Windows zum Entwickeln nutzen.**
- **Compiler und Entwicklungsumgebung installieren und nutzen.**
- **Eigene Code-Projekte unter Versionsverwaltung stellen.**

In Abbildung 2.1 kannst du schon einmal die Entwicklungsumgebung sehen, die du im Verlauf dieses Kapitels auf deinem Rechner installieren wirst.

zum Heft-Forum

Die hier beschriebenen Installationen werden auf den meisten Computern funktionieren. Trotzdem wird es hin und wieder zu Fehlermeldungen kommen. Wichtig ist dann, dass du dich davon nicht abschrecken lässt und versuchst, das Problem selbst zu lösen - denn auch das gehört zum Programmieren dazu!

Die Installation von Entwicklungs-Tools auf deinem eigenen Rechner wird ein wenig dauern. Da jeder Computer etwas anders ist (Betriebssystem, installierte Programme, Hardware), kann es unter Umständen zu Fehlermeldungen kommen, die hier im Kurs nicht beschrieben sind.

Lass dich davon nicht frustrieren, sondern wirf einen Blick ins Forum. Vielleicht findest du ja dort schon eine Lösung. Falls nicht, dann schreib einen eigenen Eintrag.

3.4 Teil 1: Ver- oder Entschlüsselung auswählen

3.4.1 Ablaufdiagramm zum Programmstart

In diesem Modul wirst du dein erstes Programm entwickeln, in dem unter anderem if-else-Verzweigungen und while-Schleifen eine wichtige Rolle spielen.

Ziel des Programms wird es sein, einen vom Benutzer eingegebenen Text nach einem bestimmten Schema in eine Geheimbotschaft zu verschlüsseln. Mit einer Verzweigung soll abgefragt werden, ob der Text ver- oder entschlüsselt werden soll und mit einer Schleife werden dann alle Buchstaben im eingegebenen Text nacheinander encodiert oder decodiert.

Wenn das Programm fertig ist und alles funktioniert, dann kannst du damit z.B. diese Geheimbotschaft entschlüsseln:

```
Tvqfs"!Ev!ibtu!efo!Dpef!hflobdlu"
```

Das Programm lässt sich in mehrere Blöcke einteilen, die nacheinander ausgeführt werden. Den ersten Teil bis zur Eingabe des zu verschlüsselnden Texts kannst du im Ablaufplan in Abbildung 3.2 sehen.

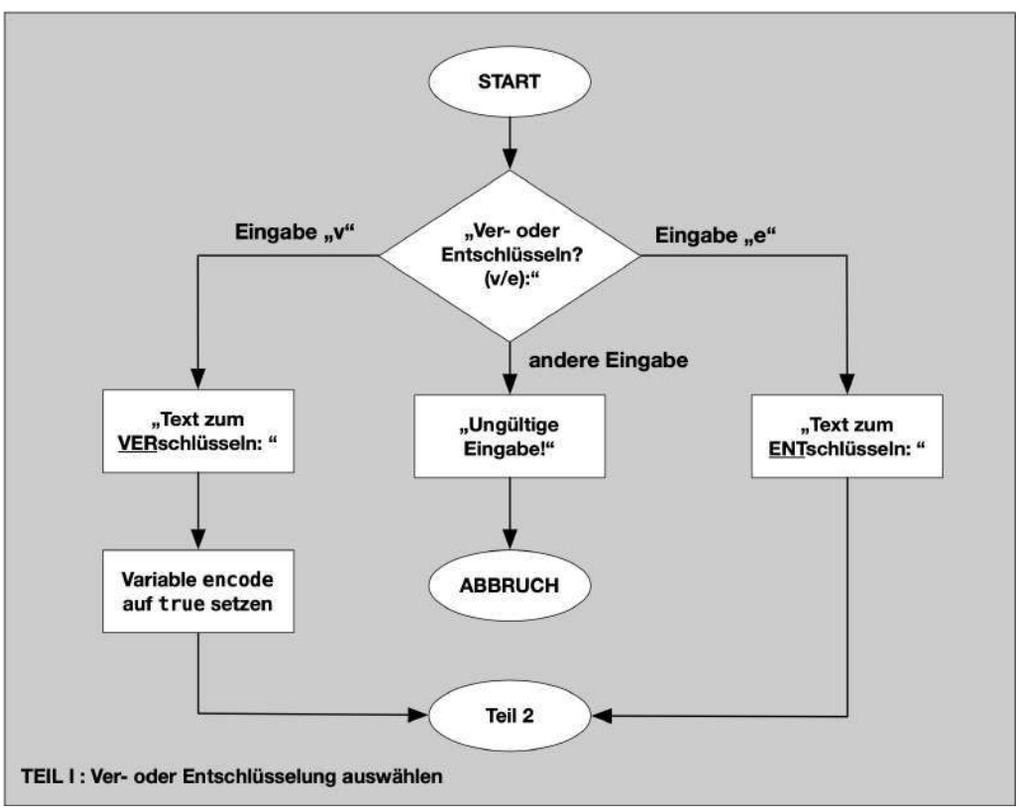


Abbildung 3.2: Ablaufplan Text-Verschlüsselung (Teil 1)

Direkt nach Programmstart wird als erstes abgefragt, ob VER- oder ENTschlüsselt werden soll.

Im ersten Fall verläuft der Programmfluss im Diagramm durch den linken Pfad und es wird ein entsprechender Text ausgegeben und eine Statusvariable namens encode auf den Wahrheitswert true gesetzt. Wahrheitswerte werden in C++ übrigens mit dem Datentyp bool gespeichert (dazu später mehr), der nur die beiden Zustände „wahr“ oder „falsch“ kennt und sich damit gut für Bedingungen und logische Verknüpfungen eignet.

Wenn der Benutzer lieber ENTschlüsseln möchte, dann bleibt der Wert von encode stattdessen auf false und es wird im rechten Pfad ein etwas anderer Text ausgegeben. In beiden Fällen geht es

Der Quellcode in Listing 3.9 wurde so erweitert, dass in **Zeile 12** die Variable encode standardmäßig den Wahrheitswert false erhält. Nur, wenn bei der Abfrage in **Zeile 10** der Buchstabe «v» eingegeben wurde, wird er im Inneren des neu definierten Blocks (**Zeilen 14 – 17**) auf den Wert true gesetzt.

```
6  int main()
7  {
8      string input;
9      cout << "Ver- oder Entschlüsseln? (v/e):" << endl;
10     getline(cin, input);
11
12     bool encode = false; // später ENTschlüsseln
13     if (input == "v")
14     {
15         cout << "Text zum VERschlüsseln: " << endl;
16         encode = true; // später VERschlüsseln
17     }
18     cout << "encode = " << encode << endl;
19
20     return 0;
21 }
```

Code

Listing 3.9: Mehrere Anweisungen in Blöcken gruppieren

Später werden wir den Wert von encode nutzen, um den Programmfluss in Abhängigkeit vom ausgewählten Modus zu verzweigen.

Wenn du den Code ausführst und den Buchstaben «v» eingibst, dann sieht die Ausgabe folgendermaßen aus:

```
Ver- oder Entschlüsseln? (v/e): v
Text eingeben:
encode = 1
```

Wie du an der Ausgabe sehen kannst, wurde die Bedingung der if-Anweisung erfüllt und die Anweisungen im nachfolgenden Block ausgeführt.

Übung 3.4.1. *Finde heraus, was passiert, wenn der Buchstabe «e» oder ein anderes Zeichen eingegeben wird.*

Hinweis: Im Online-Editor auf der Lernplattform erfolgt die Eingabe über ein Textfeld oberhalb des Quellcodes. Wenn du den Code in deiner eigenen Entwicklungsumgebung ausführst, dann erfolgt die Ein- und Ausgabe von Text über das Terminal.

1-2x

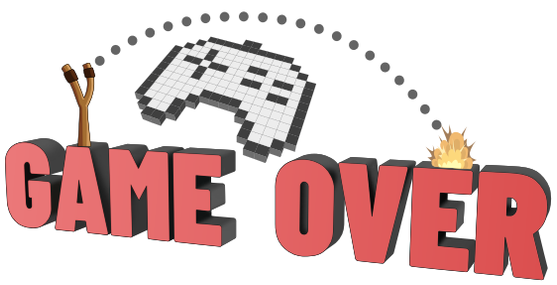
3.4.3 Entschlüsselung auswählen mit IF-ELSE

Mit dem Code aus dem letzten Abschnitt können wir zwar mehrere Anweisungen gruppieren und von einer Bedingung abhängig machen (Taste «v» gedrückt), aber eine richtige Verzweigung haben wir noch nicht programmiert.

In diesem Abschnitt soll es aus diesem Grund nun darum gehen, auch für den Fall, dass die Bedingung in **Zeile 13** von Listing 3.9 nicht erfüllt ist, bestimmte Anweisungen auszuführen und auf diese Weise den Programmfluss zu verzweigen.

In C++ gibt es für diesen Zweck das Schlüsselwort else, das zusammen mit if verwendet wird. Damit können wir den Programmfluss so umlenken, dass Anweisungen nur dann ausgeführt werden, wenn die mit if geprüfte Bedingung falsch ist. Die Struktur einer derartigen Verzweigung sieht folgendermaßen aus:

5 Projekt 3: Wütende Vögel im Terminal



Einführung

5.1 Worum geht es in diesem Kapitel

GitHub-Repo

Herzlich willkommen zum dritten und letzten Projekt in diesem Kurs. Du hast mit den ersten beiden Projekten schon einige Einblicke in die Programmierung mit C++ bekommen und bist in der Lage, erste Programme zu schreiben.

Beim Projekt „Geheime Nachrichten“ und auch beim „Player für ASCII-Filme“ handelte es sich um *prozeduralen* Code, d.h. die Anweisungen werden nacheinander abgearbeitet.

In diesem Kapitel wollen wir jetzt einen Schritt weiter gehen und die *objektorientierte Programmierung (OOP)* einführen. Dabei handelt es sich um eines der wichtigsten Konzepte von C++ (und vielen anderen Sprachen), bei der Daten und Funktionen zu *Objekten* zusammengefasst werden. Dadurch wird das Entwickeln deutlich vereinfacht, besonders wenn es sich um größeren Programme handelt.

Das kannst du nach diesem Kapitel:

- **Adressen von Variablen auslesen und in *Zeigern* speichern.**
- ***Funktionen* schreiben und Daten austauschen.**
- ***Zusammengesetzte Datentypen* erstellen.**
- **Mit *objektorientiertem Code* komplexere Programme entwickeln.**

Als Kapitel-Projekt wirst du mit Hilfe der OOP ein Spiel entwickeln, das an den Klassiker „Angry Birds“ angelehnt ist und bei dem Vögel mit einer Schleuder auf Schweine abgefeuert werden.

Wie bisher auch schon lernst du im Grundlagenteil die nötigen Basics, um den Projekt-Code zu verstehen. Wenn du damit fertig bist, wird die Entwicklung des Spiels in drei Schritten ablaufen:

5.2.1 Zeiger und Referenzen


Zeiger und Referenzen

Wenn du in einem Programm eine Variable definierst und mit einem Wert initialisierst, dann wird dieser im Speicher deines Computers an eine bestimmte Stelle geschrieben. Die Lage des Wertes im Speicher wird als *Adresse* bezeichnet, mit der du aber beim normalen Gebrauch einer Variablen in der Regel nichts zu tun hast. Stattdessen erhalten Variablen einen Namen, unter dem die Daten im Speicher zu erreichen sind.

Im folgenden Beispiel sind zwei Variablen unterschiedlichen Typs zu sehen, die direkt bei der Definition mit Werten initialisiert werden:

```
1  // Definition von Variablen
2  int i{42};
3  string s{"Hallo"};
```

Listing 5.1: Schon bekannt: Initialisierung von Variablen

Der Name der Variablen (auch *Identifier* genannt) hilft dir dabei, auf die Speicheradresse zuzugreifen, ohne dafür eine kryptische Adressbezeichnung nutzen zu müssen. Du kannst z.B. mit dem Befehl `i=23` einen neuen Zahlenwert an die für `i` reservierte Stelle in den Speicher schreiben.

Manchmal ist es aber sinnvoll, neben dem ursprünglichen Variablennamen einen weiteren Identifier zu haben, unter dem auf die Daten im Speicher zugegriffen werden kann. Zwei typische Beispiele aus der Praxis sind:

1. Rückgabe von Daten aus einer Funktion heraus, ohne den *return*-Mechanismus zu nutzen.
2. Speichern von Adressen in einer Liste, z.B. um die Vorteile eines Vektors zu nutzen, ohne eine Kopie der darin gespeicherten Daten anzulegen.

Bevor wir uns beide Beispiele in der Praxis ansehen, musst du aber zunächst wissen, was es mit dem *Adress-Operator* und mit dem *Dereferenzierungs-Operator* auf sich hat.

Der Adress-Operator &

Wenn du herausfinden möchtest, wie die Adresse einer Variablen lautet, dann hilft dir dabei der Dereferenzierungsoperator `&`, den du wie in **Zeile 4** von Listing 5.2 gezeigt einfach vor den Variablennamen schreiben musst.

```
1  // Ermittlung der Speicheradresse
2  int x{5};
3  cout << x << endl;  // Wert
4  cout << &x << endl; // Speicheradresse
```

Listing 5.2: Speicheradresse mit & ausgeben


Code

Wenn du den Code ausführst, dann sieht die Ausgabe dazu wie folgt aus:

```
5
0x7ff7b735f4cc
```

In der ersten Zeile ist, wie zu erwarten, der Wert der Variablen «x» zu sehen. Und in der zweiten Zeile findest du die dazugehörige Adresse im Speicher deines Rechners. Die Zeichen «0x» am Anfang geben übrigens an, dass die Adresse als Hexadezimalzahl ausgegeben wurde (d.h. zur Basis 16).

5.3 Verständnisfragen und Aufgaben

In diesem Abschnitt findest du wie auch schon in den Kapiteln 3.3 und 4.3 einige Fragen und Aufgaben, mit denen du dein Wissen zu C++ testen kannst.

Mein Tipp: Schau erst dann in die Lösungen im Anhang, wenn du nicht mehr weiter kommst, denn dann ist der Lernerfolg am größten!

5.3.1 Verständnisfragen

zu Zeiger & Referenzen

Frage 5.3.1. *Was ist eine Referenz auf eine Variable?*

a) *Die Adresse der Variablen im Speicher.*
b) *Der Datentyp der Variablen.*
c) *Eine Kopie der Variablen.*

Frage 5.3.2. *Schau dir den nachstehenden Code genau an.*
Wird durch die Anweisung `p_a++` *der Wert von* `a` *hochgezählt?*

```
int a{0};
int *p_a = &a;
p_a++;
```

a) *Ja*
b) *Nein*

Frage 5.3.3. *Schau dir den nachstehenden Code genau an.*
Was geschieht, wenn das Programm ausgeführt wird?

```
void Verdopple(int &b)
{
  b *= 2;
}

int main()
{
    int b{2};
    Verdopple(b);
    return 0;
}
```

a) *Die Variable* `b` *in* `main` *wird verdoppelt.*
b) *Der Compiler meldet einen Fehler, da* `b` *sowohl in* `main()` *als auch in der Funktion* `Verdopple()` *definiert ist.*
c) *Die Variable* `b` *wird nur innerhalb von* `Verdopple()` *mit 2 multipliziert.*
d) *Der Compiler meldet einen Fehler, da der Adressoperator an der falschen Stelle steht.*

Frage 5.3.4. *Ist die folgende Aussage richtig oder falsch?*

„Mit einem Zeiger wird der Typ und die Adresse einer Variablen repräsentiert.“

a) *Richtig*
b) *Falsch*

Frage 5.3.5. *Welche Zuweisung an die Variable* `int *p_i` *ist zulässig?*

a) *Eine Variable vom Typ* `int`.
b) *Eine Ganzzahl.*
c) *Die Adresse einer Variablen vom Typ* `int`.
d) *Die Adresse jeder beliebigen Variablen.*