Modul 319 Prüfung II Theorie Zusammenfassung

**Daten einlesen:** **scanf** ist das Gegenstück zu **printf**, um Daten von der Konsole einzulesen.

Der Syntax von **scanf** ist ähnlich wie bei **printf:**

Ein Bild, das Text, Schrift, weiß, Design enthält.

Automatisch generierte BeschreibungDas kaufmännische „Und“ wird hier als Adressoperator bezeichnet und bewirkt, dass der Wert in der Konsole an die Adresse der Variablen a im Speicher geschrieben wird.

**Formatzeichen**

Wie bei **printf** müssen Sie bei **scanf** die korrekten Formatzeichen verwenden. Um zum Beispiel in eine **double** Variable einzulesen wird **%lf** benötigt

Ein Bild, das Text, Schrift, weiß, Screenshot enthält.

Automatisch generierte Beschreibung

**Programmaufbau**

Ein Programm wird immer von oben nach unten bearbeitet. Die einzelnen Anweisungen werden als Sequenzen bezeichnet. Die Sequenzen müssen in einer bestimmten Reihenfolge ausgeführt werden. Beispielsweise macht es keinen Sinn eine Berechnung auszuführen, bevor Daten eingelesen wurden. **DEVA: Ein Programm muss immer nach dem DEVA-Prinzip aufgebaut sein.**

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

**Inkrement und Dekrement Operationen**

Um gewisse Berechnungen abzukürzen, lassen sich das Inkrement und Dekrement Operationen anwendenEin Bild, das Text, Screenshot, Schrift, Algebra enthält.

Automatisch generierte Beschreibung

**Mathematikbibliothek**

Mit Hilfe der Mathematikbibliothek können weitere Funktionen wie Wurzelziehen oder Potenzieren verwendet werden. Die Mathematikbibliothek muss mit ***#include <math.h>***eingebunden werden. Die sogenannte Signatur einer Funktion ist von grosser Bedeutung. Mit der Kenntnis der Signatur weiss man wie eine bestimmte Funktion angewendet werden muss. **Die Signatur stellt damit eine Vorlage für die Anwendung dar.**

**Wurzel**

Die Signatur der Wurzelfunktion lautet ***double sqrt(double x)*** Was bedeutet dies nun?

***sqrt***: Das ist der Name der Funktion, mit diesen Namen wird die Funktion aufgerufen.

***double x***: Die Funktion benötigt einen Parameter innerhalb der runden Klammern. Der Parameter muss vom typ double sein.

***double***: Das Resultat der Berechnung ist ebenfalls vom Typ double und muss einer Wertzuweisung an einer Variable vom Typ double gereicht werden.

Ein Bild, das Text, Schrift, Reihe, Screenshot enthält.

Automatisch generierte Beschreibung

**Potenz**

Signatur der Potenzfunktion: ***double pow( double b, int e)*** Damit ist folgende Anwendung korrekt

Ein Bild, das Text, Schrift, Screenshot, Algebra enthält.

Automatisch generierte Beschreibung

**Selektionen**

**If…else..**

Bei einer Verzweigung des Programmablaufs werden bestimmte Codebestandteile (Anweisung(en) 1) nur ausgeführt, wenn eine bestimmte Bedingung erfüllt ist. Ist die Bedingung nicht erfüllt, können alternativ andere Codebestandteile (Anweisung(en) 2) ausgeführt werden. In beiden Fällen fährt das Programm mit – von der Bedingung unabhängigen – weiteren Anweisungen (Anweisung(en)3) fort:

Ein Bild, das Text, Schrift, Rechteck, Screenshot enthält.

Automatisch generierte Beschreibung

Eine Verzweigung wird als if…else.. Statement implementiert

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

**If ..ohne else**

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte BeschreibungNicht immer ist es nötig einen else-Teil bei einer Verzweigung zu haben. Im folgenden Beispiel wird der Zusatz positive Zahl ausgegeben, wenn immer die Zahl grösser 0 ist. In beiden Fällen wird die Zahl ausgegeben.

**Verschachtelungen**

If...else.. Anweisungen lassen sich beliebig oft verschachteln, d.h. jedes if und/oder else können weitere if..else.. Anweisungen enthalten. Dies soll am Beispiel einer Notengebung veranschaulicht werden.

Ein Bild, das Text, Screenshot, Reihe, Schrift enthält.

Automatisch generierte Beschreibung

**Formatierregeln für Code**

Bei Verschachtelungen geht schnell die Übersicht verloren. Die Darstellung des Codes, beziehungsweise die Einrückungen müssen deshalb gewisse Regeln einhalten:

1. Die öffnende und schliessende Blockklammern stehen links an der Vorgängerzeile ausgerichtet.
2. Der Code innerhalb der öffnenden und schliessenden Blockklammern steht um 1 Tabulator nach rechts eingerückt.

Ein Bild, das Text, Screenshot, Schrift, Diagramm enthält.

Automatisch generierte Beschreibung

**Switch Case**

Verschachtelungen von Verzweigungen können sehr schnell übersichtlich werden. Mit jeder Verschachtelung rückt der Code weiter nach rechts.

Ein Bild, das Text, Reihe, Screenshot, Diagramm enthält.

Automatisch generierte Beschreibung

Eine Möglichkeit das zu umgehen ist das Verketten von Verzweigungen, das kann hinsichtlich der Performance erhebliche Nachteile nach sich ziehen. Insbesondere, wenn die Prüfungen allenfalls aufwändiger sind. Daher lässt sich die Mehrfachverzweigung in C mit einem sogenannten switch case umsetzen.

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibungswitch (Ausdruck) legt mit Ausdruck den zu vergleichenden Ausdruck, gegenüber dem die Fälle geprüft werden soll

**case <Wert>** markiert den Fall der mit dem nachfolgenden Code behandelt werden soll. Dabei wird auf **<Ausdruck> == Wert** geprüft. Trifft dies zu, wird der Code ausgeführt, ansonsten übersprungen. Das Ender der Fallbehandlung wird mit **break;** markiert. **default** markiert den Codeabschnitt, welcher ausgeführt werden soll. Abgesehen vom Vergleich ist die Funktionsweise identisch zu **case.**

**while Schleife**

C hat 3 verschiedene Schleifen, die **while, do while und for Schleife**, sogenannte Iterationsanweisungen oder auch Wiederholungen.

Syntax: Bei einer while-Schleife werden bestimmte Codebestandteile (Anweisung(en)1) so lange wiederholt als eine bestimmte Bedingung erfüllt ist. Nach jedem Schleifendurchgang wird die Bedingung erneut überprüft. Solange die Bedingung erfüllt ist, werden die Anweisung(en) 1 wiederholt. Ist die Bedingung nicht mehr erfüllt fährt das Programm mit den Anweisungen nach der Schleife fort.

Ein Bild, das Text, Diagramm, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

Die while-Schleife ist eine **kopfgesteuerte Schleife** und führt einen Block von Anweisungen so lange aus, wie die Schleifenbedingung wahr ist. Die Syntax der while-Schleife sieht folgendermassen aus:

Ein Bild, das Text, Schrift, Screenshot, weiß enthält.

Automatisch generierte Beschreibung

Struktogramm while-Schleife

Ein Bild, das Text, Screenshot, Schrift, Reihe enthält.

Automatisch generierte Beschreibung

DEVA

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

Formatzeichen

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Diagramm, Schrift enthält.

Automatisch generierte Beschreibung