

Dieses Skript unterliegt der GNU General Public License, deren Text im Internet nachzulesen ist. Das Copyleft liegt bei dem(n) Autor(en).

Ziele: höhere Programmiersprache der 3. Generation
Entwurf, Implementierung, Dokumentation, Test

Allgemeines:

- Betriebssystem Unix wurde zu 98% in C programmiert (Kernigham & Ritchie)
- C ist für nahezu alle Plattformen verfügbar: DOS, Win x.xx, Win 95/98, Unix, usw.
- C ist für die Programmierung systemnahen, technischer und betriebswirtschaftlicher DV-Verfahren einsetzbar

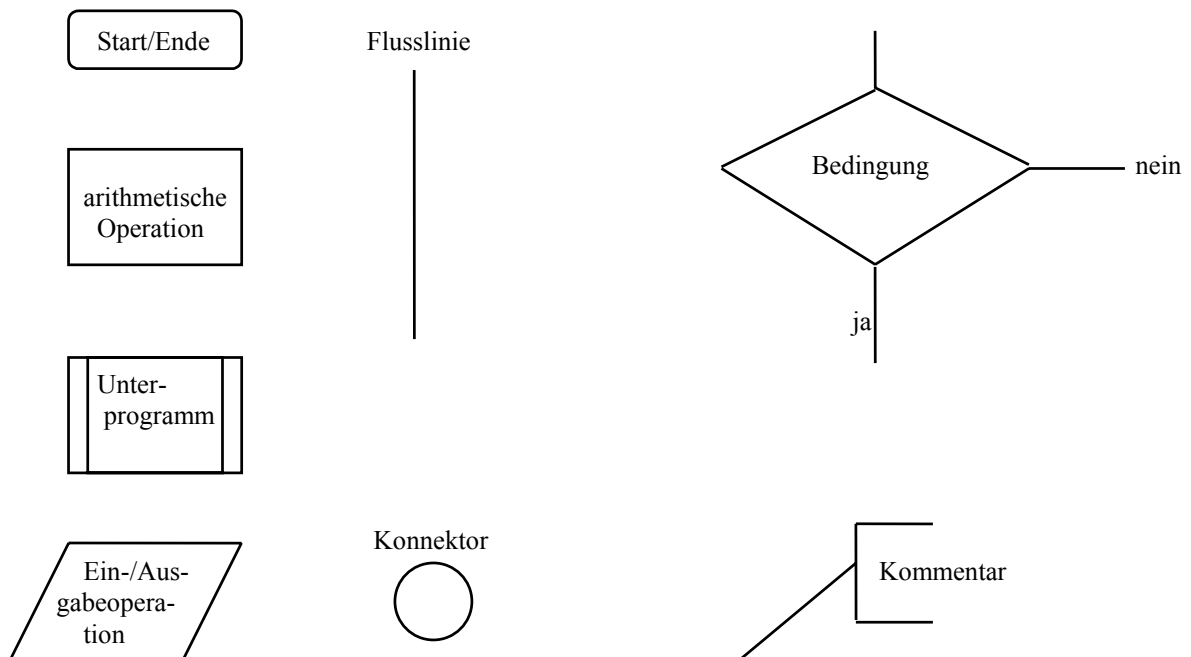
Grundlagen:

Was ist ein Programm?

John v. Neumann: Folge von Anweisungen, die sequenziell ausgeführt werden, wobei bedingte und unbedingte Sprünge möglich sind.

Goto Problem: Sprünge führen zu „Spagetti-Code“ - Programmfehler proportional zur Anzahl der Goto Anweisungen
- Programme werden schlechter wartbar

Programmablaufplan-Symbole:



Debugging:

Syntaxfehler: - werden vom Compiler erkannt und angezeigt
- können in Windows durch Doppelklick auf die Fehlermeldung angesprochen werden

Laufzeitfehler: - treten während des Programmablaufs auf und führen zum Absturz des Programmes
- z.B.: Division durch 0, fehlerhafte Parameterübergaben, ...

Logische Fehler: - führen zu falschen Ergebnissen aufgrund inkorrekt abläufe und Berechnungen

Zum finden von **Laufzeit-** und **logischen Fehlern** verwendet man einen **Debugger**. Mit ihm kann das Programm schrittweise ausgeführt, Anweisungen übersprungen und Variableninhalte überprüft/ausgewertet/geändert, Haltepunkte gesetzt/gelöscht, Ausdrücke überwacht, usw. werden.

Top-Down-Programmmentwurf

- Entwurfstechnik vom Groben zum Feinen: schrittweise Verfeinerung des Programms
- Basis: z.B. Strukturierte Analyse (SA nach Tom DeMarco 1978) der Problemstellung
 - Konzeptdiagramm mit allen Terminatoren des Problems und dem Gesamtprozess
 - Verfeinerung des Gesamtprozesses in Einzelprozesse und durch Darstellung der Datenflüsse zwischen diesen und Speichern
- Verfeinerte Prozesse sind die gesuchten Programmteile (Module)

printf

Formatangabe	Eingabewert
%d	Ganze Dezimalzahl
%i	Ganze Dezimalzahl
%u	Ganze Dezimalzahl ohne Vorzeichen
%o	Ganze oktale Zahl
%x, %X	Ganze hexadezimale Zahl
%c	ASCII-Zeichen
%f	Gleitkommazahl
%e, %E	Gleitkommazahl in Exponentialschreibweise
%g, %G	Gleitkommazahl in den Formaten %f oder %e
%s	Zeichenkette
%p	Zeiger (Adresse)
%n	Zeiger (Anzahl der schon ausgegebenen Zeichen)

Das Prozentzeichen selbst kann mit %% ausgegeben werden.

scanf

Formatangabe	Eingabewert
%d	Ganze Dezimalzahl vom Typ int
%hd	Ganze Dezimalzahl vom Typ short
%ld	Ganze Dezimalzahl vom Typ long
%i	Ganze Zahl von Typ int (Eingabe dezimal, oktall mit führender 0 oder hexadezimal mit führendem 0x)
%u	Ganze Dezimalzahl vom Typ int ohne Vorzeichen
%hu	Ganze Dezimalzahl vom Typ short ohne Vorzeichen
%lu	Ganze Dezimalzahl vom Typ long ohne Vorzeichen
%o	Ganze okatale Zahl von Typ int
%ho	Ganze okatale Zahl von Typ short
%lo	Ganze okatale Zahl von Typ long
%x	Ganze hexadezimale Zahl vom Typ int
%hx	Ganze hexadezimale Zahl vom Typ short
%lx	Ganze hexadezimale Zahl vom Typ long
%c	ASCII Zeichen vom Typ char
%f	Gleitkommazahl vom Typ float
%lf	Gleitkommazahl vom Typ double
%Lf	Gleitkommazahl vom Typ long double
%e, %E	Gleitkommazahl vom Typ float in Exponentialschreibweise
%le, %lE	Gleitkommazahl vom Typ double in Exponentialschreibweise

%g, %G	Gleitkommazahl vom Typ float (exponential oder dezimal)
%lg, %lG	Gleitkommazahl vom Typ long double (exponential oder dezimal)
%s	Zeichenkette
%p	Zeiger
%n	Keine Benutzereingabe (zeigt die Anzahl der bis dahin eingelesenen Zeichen an)

do-Schleife (solange ausführen, solange bedingung = true (erfüllt) ist)

```
do
    {
        <anweisung>;
        ...
    }
while (<bedingung>);
```

for-Schleife (solange ausführen, solange bedingung = true (erfüllt) ist)

```
for ( [<init>] ; [<bedingung>] ; [<finit>] )
    {
        <anweisung>;
        ...
    }
```

<init> Folge von Anweisungen, die zu Beginn der Schleife einmalig ausgeführt werden
z.B.: Kontrollvariable auf Startwert setzen

<bedingung> Wird vor der Ausführung der Anweisungen geprüft, sie muss erfüllt sein, damit die Anweisungen ausgeführt werden.

<finit> Folge von Anweisungen, die jeweils nach den Ausführen der <anweisungen> innerhalb der Schleifen ausgeführt werden.

switch-Anweisung

```
switch (<variable>)
    {
        case <wert1>:
            <anweisung>;
            ...
            break;

        case <wert2>:
            <anweisung>;
            ...
            break;

        ...

        [default:
            <anweisung>;
            ...
            break;]
    }
```

Arrays

- 1-dimensionale Arrays sind lineare Datenstrukturen eines Datentyps
- Eine Gruppe von Variablen gleichen Datentyps wird unter einem Daten angesprochen
- Die einzelne Variable wird über einen Index angesprochen
- Deklaration eines eindimensionalen Arrays `<Datentyp> <ArrayName> [<AnzahlElemente>;`

Beispiel: 1-dim. Integer Array der Länge 100 `int x[100];`

strncmpi (<string1>,<string2>,int)
stricmp (<string1>,<string2>)
strnicmp (<string1>,<string2>,int)
strcoll(<string1>,<string2>

strspn(<string1>,<string2>) Strings auf Zeichen untersuchen
strcspn(<string1>,<string2>)
strchr(<string>,char)
strrchr(<string>,char)
strstr(<string1>,<string2>)
strpbrk(<string1>,<string2>)

strlwr(<string>) Strings manipulieren
strupr(<string>)
strrev(<string>)
strdup(<string>)
strset(<string>,char)
strnet(<string>,char, int)

isupper(char) Chars abfragen
islower(char)
isalnum(char)
isalpha(char)
isascii(char)
iscntrl(char)
isdigit(char)
isprint(char)
ispunct(char)
isspace(char)
isxdigit(char)

toupper(char) Chars umwandeln
tolower(char)
toascii(char)

Gefahren beim Umgang mit Strings

- Das Null-Byte am Schluß fehlt oder wird überschrieben => es werden String-Funktionen ausgeführt, bis ein zufälliger Null-Character gefunden wird.

Beispiel: `x[strlen(x)] = '\n';` überschreibt den Null-Character und läßt damit einen undefinierten String entstehen

- Das Char-Array ist zu klein dimensioniert => der String wird zum Teil in Bytes abgelegt, die zu anderen Variablen gehören oder in denen Programmteile liegen

Beispiel: `char x[] = „Adam“;`
 `char y[] = „Eva“;`
 `strcat(x, y);` zerstört die Bytes hinter dem x-Array

printf / scanf - Erweiterungen

- Formatierte Ausgabe in einen String

```
sprintf(<zielstring>,<formatstring>,<parameterliste>);
```

- Beschränktes Einlesen in einen String

```
scanf(„ ... %<width>s ...“, ...);
```

- Invariante Ausgabe eines Strings

```
printf(„ ... -<width>.<width>s ...“, ...);
```

- Dynamisches Generieren eines Formatstrings über sprintf oder strcpy oder strcat, oder ...

Aus André Willms, C Programmierung lernen, Addison Wesley

- geschweifte Klammern fassen Anweisungen zu einem Block zusammen
- Stringkonstanten stehen immer innerhalb doppelter Anführungszeichen
- Anweisungen, denen kein Anweisungsblock folgt, werden mit einem Semikolon abgeschlossen
- Anweisung mit # am Anfang sind Präprozessoranweisungen und müssen ganz links in der Zeile stehen
- Funktionsaufrufe besitzen immer ein Paar runde Klammern hinter dem Funktionsnamen
- Funktionsnamen müssen mit einem Buchstaben oder Unterstrich beginnen und dürfen weiterhin nur Buchstaben, Ziffern oder Unterstriche enthalten
- Bei Namen wird Groß- und Kleinschreibung unterschieden; zur Unterscheidung werden nur die ersten 31 Zeichen herangezogen
- definiert = Speicherplatz wird reserviert
- initialisiert = Ordnet der Variablen einen Wert zu
- Vor dem Gleichheitszeichen dürfen keine konstanten Ausdrücke stehen
- Nicht initialisierte Variablen haben einen nicht vorhersagbaren Wert
- Vorsicht: Falsche Eingaben haben im allgemeinen keinen Laufzeitfehler zur Folge
- Fließkommazahlen werden an integer-Variablen durch Wegfall der Nachkommastellen angepasst. Es findet KEIN Auf- oder Abrunden statt
- % Modulo => bestimmt den Rest einer Division von Ganzzahlen
- Achten Sie bei der Benutzung der Zuweisungsoperatoren auf die Bindungsstärke der verwendeten Rechenoperatoren
- Steht an einer Stelle im Programm ein berechenbarer Ausdruck, kann er auch an Ort und Stelle zugewiesen werden
(printf(„%i plus %i ist gleich %i „, a, b, a+b);
- Bei den Postoperatoren (wert++) wird die Variable erst „benutzt“ und dann entsprechend inkrementiert oder dekrementiert. Die Präoperatoren (++wert) inkrementieren/dekrementieren vor der Benutzung.
- Abfrage auf Gleichheit mit == (Weil der Compiler damit den Gleichheits- vom Zuweisungsoperator unterscheiden kann
- Besteht der Anweisungsblock hinter einer Kontrollstruktur nur aus einer Anweisung, dann können die geschweiften Klammern wegegelassen werden.
- Eine Bedingung ist falsch, wenn sie den Wert 0 hat.
Eine Bedingung ist richtig, wenn sie einen Wert ungleich 0 hat.

```
=> if(!x) //Abfrage auf x gleich 0
```
- Eine Bedingung darf auch in einem berechenbaren Ausdruck stehen. Sie nimmt, je nachdem, ob sie wahr oder falsch ist, den Wert 1 oder 0 an.

- Eine Variable, die innerhalb eines Anweisungsblocks definiert wurde ist lokal und gilt nur innerhalb des Blocks
- Wird ein Anweisungsblock verlassen, werden alle lokalen Variablen, die in ihm definiert wurden gelöscht.
- Gibt es mehrere Variablen mit gleichem Namen, spricht man über den Namen immer die lokalste Variable an.
- Lokale Variablendefinitionen müssen immer die ersten Anweisungen des Blocks sein, in dem sie stehen.
- Variablen, die außerhalb einer Funktion definiert werden, sind global und können in jeder Funktion benutzt werden.
- So lokal wie möglich , so global wie nötig
- Statische Variablen werden beim Verlassen ihres Bezugsrahmens nicht gelöscht, sondern behalten ihren Wert bei.
- Statische Variablen müssen bei ihrer Definition initialisiert werden.
- Funktionsparameter sind lokale Variablen der Funktion, die mit den Übergabeparametern initialisiert sind.
- Funktionen, die einen Wert zurückgeben, repräsentieren diesen Wert mit ihrem Aufruf.
- Wird der Rückgabewert einer Funktion nicht angegeben, geht der Compiler davon aus, das ein int-Wert zurückgegeben wird.
- Prototypen müssen nur die Variablentypen der Funktionsparameter, nicht jedoch deren Variablenamen enthalten.
- Eine Funktion mit Rückgabewert muß so entworfen sein, daß sie zu jeder Bedingung mit einem `return` beendet wird.
- Mehrere Anweisungen werden innerhalb von `for` mit Kommata getrennt
- Gleichheitsoperator ist `==` ; der Zuweisungsoperator ist `=`
- Die `while` Anweisung im `do ... while` Konstrukt wird immer mit einem Semikolon abgeschlossen.
- Die `continue` Anweisung springt zum Kopf der innersten `for`-, `so`- oder `while`-Anweisung, in der sie vorkommt
- Werden zwei Variablen unterschiedlichen Typs durch einen Operator miteinander verknüpft, dann bekommt das Ergebnis den genaueren Typ der beiden Variablen.
- Die `break`-Anweisung springt hinter den innersten `do`-, `for`-, `switch`- oder `while`-Anweisungsblock, in dem sie steht.
- Es können beliebig viele `case`-Anweisungen das gleiche Sprungziel haben.
(z.B.: `case 1:`
 `case2:`
 `printf(„1 und 2 wurden gewählt“);`
- Existiert für einen Wert keine `case`-Sprungmarke, wird - wenn vorhanden - die default Sprungmarke angesprochen.
- Zeichenkonstanten stehen immer zwischen einfachen Anführungszeichen.
- Der Adreßoperator `&` liefert immer einen Vektor.
- Zeiger speichern Adressen ; Variablen speichern Werte.
- Der Name einer Funktion ohne runde Klammern repräsentiert ihre Adresse (Funktionsname ist ein Vektor).
- Feldindex beginnt mit 0 => n Elemente 0 bis n-1 Indizes.
- Der Name eines Feldes ohne eckige Klammern steht für die Startadresse des Feldes. Der Feldname ist ein Vektor.
- Der Zeiger auf eine Variable ist identisch mit dem Zeiger auf ein Feld.
- Bei Zeigern auf Felder darf bei der Benutzung des Index kein Dereferenzierungsoperator angewendet werden.

- Die Inkrement/Dekrement- Operatoren inkrementieren/dekrementieren eine Zeiger um die Größe des Variablentyps, auf den der Zeiger zeigt.
- Ein String darf mit Endeckennung nur gleich oder kleiner als das ihm beherbergende Feld sein, niemals größer.
- Der Name eines Stringfeldes mit einem Index weniger als in der Definition, repräsentiert die Adresse eines speziellen Strings.

-
- feof liefert erst dann einen wahren Wert, wenn versucht wurde, Daten zu lesen, obwohl das Dateiende erreicht ist.
 - sizeof gibt die Größe der Variablen in Bytes aus ; Klammern erhöht die Übersichtlichkeit. (z.B.: char s[20]; sizeof(s);)

-
- Der Anweisungsblock einer Struktur wird hinter der geschlossenen geschweiften Klammer mit Semikolon abgeschlossen.
 - Strukturen können genauso mit Hilfe des Zuweisungsoperators = kopiert werden wie die elementaren Datentypen.
 - Strukturen besitzen fast alle Vorzüge elementarer Datentypen.
 - Während bei einer Struktur jedes Element seinen eigenen Speicherplatz bekommt, benutzen bei einer union alle Elemente denselben Speicherplatz. Die Größe des Speicherplatzes richtet sich nach dem größten Element der union.

verkettete Listen:

- Sie sollten Nutzdaten immer in einer eigenen Struktur kapseln und diese dann in die Kontrolldatenstruktur einbetten.
- Durch das Einfügen von neuen Elementen an der richtigen Stelle innerhalb der Liste, bleibt die Liste stets sortiert.

-
- Backtracking ist ein einfaches Verfahren, alle Möglichkeiten einer Situation durchzuspielen.

-
- Bei doppelten Anführungszeichen sucht die #include Direktive die Datei im aktuellen Arbeitsverzeichnis.
 - srand mit time initialisiert macht rand zum „echten zeitabhängigen“ Zufallsgenerator.

Aus André Willms, C++ Programmierung lernen, Addison Wesley

- Variablendefinitionen werden mit einem Semikolon abgeschlossen
- setw, setfill, left, right, oct, hex, showbase, uppercase, ... => Manipulatoren (für z.B.: cout)
- setw wirkt sich nur auf die unmittelbar folgenden Ausgabe aus.
- Beutzen sie zum Inkrementieren/Dekrementieren die Operatoren ++/-- und zum Addieren/Subtrahieren die Operatoren +/-
bzw. += / -=
- Ein Prototyp einer Funktion sieht aus wie der Funktionskopf ohne Variablenamen mit abschließendem Semikolon.
- Der Bereich eines Programms, in dem eine bestimmte Variable existent ist, nennt man Bezugsrahmen der Variablen.
- Ein Funktionsname ist dann überladen, wenn mehrere Funktionen diesen Namen besitzen.
- Funktionen mit gleichem Namen und gleichen Bezugsrahmen müssen eine unterschiedliche Parameterliste besitzen.

- Nur die Funktionsparameter am Ende der Parameterliste dürfen default-Werte besitzen.
- Eine Referenz ist keine Kopie, sondern steht als Synonym für die Originalvariable.
- Referenztypen müssen bei ihrer Definition initialisiert werden.
- Das Schlüsselwort inline ist nur eine Empfehlung. Es ist für den Compiler nicht bindend.
- Das Prüfen auf Gleichheit wird mit ZWEI Gleichheitszeichen (==) vorgenommen.
Eine Zuweisung nur mit einem (=).