

## Operatoren in C

|                          |                         |
|--------------------------|-------------------------|
| Zuweisen                 | =                       |
| Addition, Subtraktion    | +, -                    |
| Multiplikation, Division | *, / -> Div. Ohne Rest! |
| Modulo-Divisor           | % nur Rest              |
| In- /bzw. dekrementieren | a++, a--                |
| Größer als               | a>b                     |
| Kleiner als              | a<b                     |
| Gleich                   | a==b -> ! 2x gleich!    |
| Nicht gleich             | a!=b                    |
| Kleiner oder gleich      | a<=b                    |
| Größer oder gleich       | a>=b                    |

Logische Operatoren: !a -> NOT a, a&&b -> a AND b, a||b -> a OR b

-> Vergleichs- und logische Operatoren: Benutzung z.B. als Bedingung

-> +=, /=, %=, &= usw. sind möglich, Variable wird verarbeitet und zugeordnet: bsp.: a=a+b -> a+=b

Bit-Operatoren:

### Ein- und Ausgabe in C:

Ausgabe: `printf(«Text %i %\n», 4, 8);`

\n fängt eine neue Linie an

- Eine Zahl nach dem Prozentzeichen spezifiziert die Mindestanzahl Stellen der Zahl
- z.B. `%3.2f` -> float mit mindestens 3 Stellen, 2 nach dem Komma

Eingabe: `scanf(«%i», &zahl);`

Erwartet ein integer, setzt es in die Variable zahl

### Programmaufbau in C

`/** Kommentar, * in jeder Zeile, bis */`

`-> Präprozessor:`

`#include <stdio.h>`

`#include <weitere C-Bibliotheken>`

*funktionen (primitiv oder ganze)*

`int main (void) {`

`code;`

`return 0;`

`}`

### Entscheidungen: if...else und ? im printf

- Mit ? Entscheidung direkt im printf

`erfuellt=1;`

`printf(«%s \n», erfuehlt ? «wahr» : «falsch»)`

- Mit if...else

`If (erfuellt=1){`

`printf(«wahr \n»)`

`} else {`

`printf(«falsch \n») }`

### Entscheidungen: switch

- Mehrfachauswahl wird vereinfacht
- !Break unbedingt nötig!

`int day = 3;`

`switch (day) {`

`case 1:`

`printf(«Montag»);`

`break;`

`case 2:`

`printf(«Dienstag»);`

`break;`

`case 3:`

`case 4:`

`printf(«Mittwoch oder Donnerstag»);`

`break;`

`usw.`

`default: <- nötig, für unvorhergesehene Zustände`

`printf(«kein Tag»); }`

### Schleifen: for-Schleife

- Gut für exakte Anzahl Wiederholungen

`int main (void) {`

`int i;`

`for (i=1; i<10; i++){`

`printf (« Zeile %d \n », i);`

`}`

`Return 0; }`

### Schleifen: While-Schleife

- Block wird ausgeführt, wenn Bedingung erfüllt

`while (Bedingung) {`

`Anweisung;`

`}`

### Schleifen: Do While-Schleife

- Block wird ausgeführt, danach geprüft ob nochmal ausführen

`do {`

`Block zum Ausführen;`

`} while (Bedingung);`

`Weiter im code;`

### Typenumwandlung

- Wandelt datentypen um z.B. int <-> float

`int i=5;`

`float f = i; <-implizite Umwandlung`

`float pi = 3.1415f;`

`int i = (int) pi; <- explizite Umwandlung`

## Arrays und Zeiger:

Array-aufbau: `int a [10];`

-> a ist ein Array mit 10 Elementen vom Typ int (Index geht von 0 bis 9)!

Kann auch direkt initialisiert werden: `int b [5] = {0,1,2,3,4};`

Nachträglich eintragen: `b [0]= 5;`

-> b wird zu [5,1,2,3,4]

## Zeiger

& weist einer Variable die Adresse einer anderen zu

\* gibt den wert, worauf der Zeiger zeigt

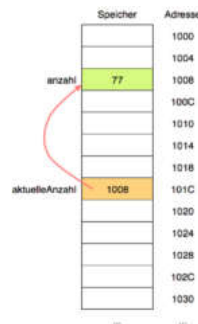
`int anzahl;`

`int *aktuelleanzahl;`

`aktuelleanzahl = &anzahl;`

- Zugriff über Zeiger

`*aktuelleanzahl = 77;`



## Zeigerarithmetik

-> zeigt ein Zeiger auf ein array, wird der Erste eintrag ausgewählt.

-> wird der Zeiger inkrementiert, wird der nächste Eintrag im Array gewählt

## Strings als Arrays

- Strings sind arrays, endet mit \0 ("x0")

`char meldung [100] = «Achtung»` <- 99 zeichen

`char fehler [ ] = «Fehler»;` <- länge 7 (6z+\0)

- nachträglich:

`strcpy (meldung, «Achtung»);`

- Ausgabe im printf mit %s

- Einlesen mit fgets

`char buf [5];`

`fgets (buf, 5, stdin);`

## Funktionen: Initialisierung

- Funktion initialisieren: vor main schreiben
- Alternativ: Funktionsprimitiv vor main
- Auf Lokale / Globale variablen achten!

`int beispiel (int wertA, int wertB);`

`int main (void) {`  
`}`

`int beispiel (int wertA, int wertB) {`  
`return xyz; }`

## Funktionen: abrufen/ausführen

`int f (int wert) {`

`wert *=2;`

`return wert; }`

`int main (void){`

`int start=3, neu=0;`

`neu= f (start);` <-Funktion wird aufgerufen

`return 0; }`

## Arrays in Funktionen

- Arrays mit Funktionen bearbeiten

`void demo (int daten [ ]);`

`daten [1] =99;`

`return; }` <- Array wird direkt bearbeitet, return nichts

`int main (void) {`

`int werte [3]={1,2,3};`

`demo (werte);`

`return 0; }`

## Bit-Operationen

|    |                           |                                    |
|----|---------------------------|------------------------------------|
| &  | Bit-AND                   | 6 & 3 -> 2 (0110)&(0011) -> (0010) |
|    | Bit-OR                    | 6   3 -> 7 (0110) (0011) -> (0111) |
| ^  | Bit-XOR                   | 6 ^ 3 -> 5 (0110)^(0011) -> (0101) |
| ~  | Bit-Negation              | ~1 -> -2 ~(0001) -> (1110)         |
| << | Bits nach links schieben  | 6<<3 -> 48 (0110)<<3 ->(0110000)   |
| >> | Bits nach rechts schieben | 9>>3 -> 1 (1001)>>3 ->(0001)       |

- Bit (auf 1) setzen: Bit-OR

`unsigned int setBit (unsigned int n, unsigned int data) {`

`int result = 0;`

`unsigned int bitn = 1 << n;`

`result = data | bitn;`

`return result; }`

- Bit löschen: Bit-AND und Bit-NOT

`unsigned int clearBit (unsigned int n, unsigned int data) {`

`int result = 0;`

`unsigned int bitn = 1 << n;`

`result = data & ~bitn;`

`return result; }`

- Bit umschalten: Bit-XOR

`unsigned int toggleBit (unsigned int n, unsigned int data) {`

`int result = 0;`

`unsigned int bitn = 1 << n;`

`result = data ^ bitn;`

`return result; }`

- Bit abtasten:

`unsigned int testBit (unsigned int n, unsigned int data) {`

`int result = 0;`

`unsigned int bitn = 1 << n;`

`if ((data & bitn) == bitn){`  
`result=1;`

`}else result =0;`

`return result; }`

## Funktionen: Testen

- Zum Testen: «expect.h» im Präprozessor

In main-Funktion:

```
expect(funktionsname(in1, in2)=erw. Resultat);
```

## Datenstrukturen: struct

- Fasst mehrere Variablen zusammen

```
Int main (void) {  
    struct Complexnumber x;  
    x.real = 7.0;  
    x.imag = 6.4;  
    printf(«(%f, %f)\n», x.real, x.imag);  
    return 0;}
```

## Datenstrukturen: enum

- Weist Wörtern eine Zahl zu.

Element 1:0, E2:1, E3: 2 usw.

```
enum Frucht {apfel, birne, zitrone};
```

## Datenstrukturen: typedef

- struct werden umbenannt

```
struct Datum_s { int tag, monat, jahr;};  
typedef struct Datum_s Datum;
```

- Datum\_s wird zu Datum

## 2D-Arrays

- Speichern von Werten

```
Int main (void) {  
    int matrix [3] [5] = {  
        {0, 1, 2, 3, 4},  
        {10, 11, 12, 13, 14},  
        {20, 21, 22, 23, 24}  
    };  
    Printf (« %d \n », matrix [2] [4] ); -> gibt 24 aus.  
    Return 0 ;}
```

## ASCII-Tabelle

| ASCII Printing Characters Chart |     |     |           |         |     |     |           |         |     |     |           |
|---------------------------------|-----|-----|-----------|---------|-----|-----|-----------|---------|-----|-----|-----------|
| Decimal                         | Hex | Oct | Character | Decimal | Hex | Oct | Character | Decimal | Hex | Oct | Character |
| 32                              | 20  | 040 | space     | 64      | 40  | 100 | @         | 96      | 60  | 140 | a         |
| 33                              | 21  | 041 | !         | 65      | 41  | 101 | A         | 97      | 61  | 141 | a         |
| 34                              | 22  | 042 | "         | 66      | 42  | 102 | B         | 98      | 62  | 142 | b         |
| 35                              | 23  | 043 | #         | 67      | 43  | 103 | C         | 99      | 63  | 143 | c         |
| 36                              | 24  | 044 | \$        | 68      | 44  | 104 | D         | 100     | 64  | 144 | d         |
| 37                              | 25  | 045 | %         | 69      | 45  | 105 | E         | 101     | 65  | 145 | e         |
| 38                              | 26  | 046 | &         | 70      | 46  | 106 | F         | 102     | 66  | 146 | f         |
| 39                              | 27  | 047 | '         | 71      | 47  | 107 | G         | 103     | 67  | 147 | g         |
| 40                              | 28  | 050 | (         | 72      | 48  | 110 | H         | 104     | 68  | 150 | h         |
| 41                              | 29  | 051 | )         | 73      | 49  | 111 | I         | 105     | 69  | 151 | i         |
| 42                              | 2A  | 052 | *         | 74      | 4A  | 112 | J         | 106     | 6A  | 152 | j         |
| 43                              | 2B  | 053 | +         | 75      | 4B  | 113 | K         | 107     | 6B  | 153 | k         |
| 44                              | 2C  | 054 | ,         | 76      | 4C  | 114 | L         | 108     | 6C  | 154 | l         |
| 45                              | 2D  | 055 | -         | 77      | 4D  | 115 | M         | 109     | 6D  | 155 | m         |
| 46                              | 2E  | 056 | .         | 78      | 4E  | 116 | N         | 110     | 6E  | 156 | n         |
| 47                              | 2F  | 057 | /         | 79      | 4F  | 117 | O         | 111     | 6F  | 157 | o         |
| 48                              | 30  | 060 | 0         | 80      | 50  | 120 | P         | 112     | 70  | 160 | p         |
| 49                              | 31  | 061 | 1         | 81      | 51  | 121 | Q         | 113     | 71  | 161 | q         |
| 50                              | 32  | 062 | 2         | 82      | 52  | 122 | R         | 114     | 72  | 162 | r         |
| 51                              | 33  | 063 | 3         | 83      | 53  | 123 | S         | 115     | 73  | 163 | s         |
| 52                              | 34  | 064 | 4         | 84      | 54  | 124 | T         | 116     | 74  | 164 | t         |
| 53                              | 35  | 065 | 5         | 85      | 55  | 125 | U         | 117     | 75  | 165 | u         |
| 54                              | 36  | 066 | 6         | 86      | 56  | 126 | V         | 118     | 76  | 166 | v         |
| 55                              | 37  | 067 | 7         | 87      | 57  | 127 | W         | 119     | 77  | 167 | w         |
| 56                              | 38  | 070 | 8         | 88      | 58  | 130 | X         | 120     | 78  | 170 | x         |
| 57                              | 39  | 071 | 9         | 89      | 59  | 131 | Y         | 121     | 79  | 171 | y         |
| 58                              | 3A  | 072 | :         | 90      | 5A  | 132 | Z         | 122     | 7A  | 172 | z         |
| 59                              | 3B  | 073 | ;         | 91      | 5B  | 133 | [         | 123     | 7B  | 173 | {         |
| 60                              | 3C  | 074 | <         | 92      | 5C  | 134 | \         | 124     | 7C  | 174 |           |
| 61                              | 3D  | 075 | =         | 93      | 5D  | 135 | ]         | 125     | 7D  | 175 | }         |
| 62                              | 3E  | 076 | >         | 94      | 5E  | 136 | ^         | 126     | 7E  | 176 | ~         |
| 63                              | 3F  | 077 | ?         | 95      | 5F  | 137 | _         | 127     | 7F  | 177 | DEL       |