| Dez | Hex | | Dez | Hex | | Dez | Hex | | Dez | Hex | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *0* | 0x00 | NUL | *32* | 0x20 | SP | *64* | 0x40 | @ | *96* | 0x60 | ` |
| *1* | 0x01 | SOH | *33* | 0x21 | ! | *65* | 0x41 | A | *97* | 0x61 | a |
| *2* | 0x02 | STX | *34* | 0x22 | " | *66* | 0x42 | B | *98* | 0x62 | b |
| *3* | 0x03 | ETX | *35* | 0x23 | # | *67* | 0x43 | C | *99* | 0x63 | c |
| *4* | 0x04 | EOT | *36* | 0x24 | $ | *68* | 0x44 | D | *100* | 0x64 | d |
| *5* | 0x05 | ENQ | *37* | 0x25 | % | *69* | 0x45 | E | *101* | 0x65 | e |
| *6* | 0x06 | ACK | *38* | 0x26 | & | *70* | 0x46 | F | *102* | 0x66 | f |
| *7* | 0x07 | BEL | *39* | 0x27 | ' | *71* | 0x47 | G | *103* | 0x67 | g |
| *8* | 0x08 | BS | *40* | 0x28 | ( | *72* | 0x48 | H | *104* | 0x68 | h |
| *9* | 0x09 | TAB | *41* | 0x29 | ) | *73* | 0x49 | I | *105* | 0x69 | i |
| *10* | 0x0A | LF | *42* | 0x2A | * | *74* | 0x4A | J | *106* | 0x6A | j |
| *11* | 0x0B | VT | *43* | 0x2B | + | *75* | 0x4B | K | *107* | 0x6B | k |
| *12* | 0x0C | FF | *44* | 0x2C | , | *76* | 0x4C | L | *108* | 0x6C | l |
| *13* | 0x0D | CR | *45* | 0x2D | - | *77* | 0x4D | M | *109* | 0x6D | m |
| *14* | 0x0E | SO | *46* | 0x2E | . | *78* | 0x4E | N | *110* | 0x6E | n |
| *15* | 0x0F | SI | *47* | 0x2F | / | *79* | 0x4F | O | *111* | 0x6F | o |
| *16* | 0x10 | DLE | *48* | 0x30 | 0 | *80* | 0x50 | P | *112* | 0x70 | p |
| *17* | 0x11 | DC1 | *49* | 0x31 | 1 | *81* | 0x51 | Q | *113* | 0x71 | q |
| *18* | 0x12 | DC2 | *50* | 0x32 | 2 | *82* | 0x52 | R | *114* | 0x72 | r |
| *19* | 0x13 | DC3 | *51* | 0x33 | 3 | *83* | 0x53 | S | *115* | 0x73 | s |
| *20* | 0x14 | DC4 | *52* | 0x34 | 4 | *84* | 0x54 | T | *116* | 0x74 | t |
| *21* | 0x15 | NAK | *53* | 0x35 | 5 | *85* | 0x55 | U | *117* | 0x75 | u |
| *22* | 0x16 | SYN | *54* | 0x36 | 6 | *86* | 0x56 | V | *118* | 0x76 | v |
| *23* | 0x17 | ETB | *55* | 0x37 | 7 | *87* | 0x57 | W | *119* | 0x77 | w |
| *24* | 0x18 | CAN | *56* | 0x38 | 8 | *88* | 0x58 | X | *120* | 0x78 | x |
| *25* | 0x19 | EM | *57* | 0x39 | 9 | *89* | 0x59 | Y | *121* | 0x79 | y |
| *26* | 0x1A | SUB | *58* | 0x3A | : | *90* | 0x5A | Z | *122* | 0x7A | z |
| *27* | 0x1B | ESC | *59* | 0x3B | ; | *91* | 0x5B | [ | *123* | 0x7B | { |
| *28* | 0x1C | FS | *60* | 0x3C | < | *92* | 0x5C | \ | *124* | 0x7C | \| |
| *29* | 0x1D | GS | *61* | 0x3D | = | *93* | 0x5D | ] | *125* | 0x7D | } |
| *30* | 0x1E | RS | *62* | 0x3E | > | *94* | 0x5E | ^ | *126* | 0x7E | ~ |
| *31* | 0x1F | US | *63* | 0x3F | ? | *95* | 0x5F | _ | *127* | 0x7F | DEL |

***Header-Dateien:***

| | |
|---|---|
| #include<conio.h> | Konsolenfunktionen<br>getch(); getche() |
| #include<stdio.h> | Standardein-/ausgabe<br>printf(); scanf(); gets(); |
| #include<stdlib.h> | Libraryfunktionen<br>rand();system(); |
| #include<string.h> | Stringfunktionen<br>strcpy(); strlen(); strcat(); strcmp(); |
| #include<math.h> | mathematische Funktionen<br>(sin(); cos(); sqrt();...) |
| #include<ctype.h> | Typumwandlungsfunktionen<br>toupper(); |
| #include<time.h> | Zeitfunktionen<br>time(); |

***Grundlegende Datentypen:***

| | |
|---|---|
| char | ganze Zahlen - 1 Byte |
| short | ganze Zahlen - 2 Byte |
| int | ganze Zahlen - 4 Byte |
| float | Fließkommazahlen - 4 Byte |
| double | Fließkommazahlen - 8 Byte |
| char* | Zeiger auf char-Wert |
| short* | Zeiger auf short-Wert |
| int* | Zeiger auf int-Wert |
| float* | Zeiger auf float-Wert |
| double* | Zeiger auf double-Wert |
| *typ bezeichner*[*n*] | Datenfelder |
| *typ bezeichner*[*m*][*n*] | mehrdimensionale Datenfelder |

***Modifizierer:***

| | |
|---|---|
| const *typ* | Konstantendeklaration |
| signed *typ* | positive und negative Werte |
| unsigned *typ* | positive Werte |

***Operatoren:***

| | |
|---|---|
| *bezeichner = ausdruck* | Zuweisung |
| + | Addition |
| - | Subtraktion |
| / | Division |
| * | Multiplikation |
| % | Modulo-Division (ganzzahliger Rest) |
| ~ | bitweise Invertierung |
| & | bitweises UND |
| \| | bitweises ODER |
| << | bitweise links schieben |
| >> | bitweise rechts schieben |

***Kurzoperatoren:***

| | |
|---|---|
| += | Addition mit Zuweisung |
| -= | Subtraktion mit Zuweisung |
| *= | Multiplikation mit Zuweisung |
| /= | Division mit Zuweisung |
| ++ | Inkrementierung |
| -- | Dekrementierung |
| *(typ)ausdruck* | cast-Ausdruck(Typumwandlung) |

***Mathematische Funktionen:***

| | |
|---|---|
| sin(*ausdruck*) | berechnet den Sinus eines Winkels |
| cos(*ausdruck*) | berechnet den Cosinus eines Winkels |
| tan(*ausdruck*) | berechnet den Tangens eines Winkels |
| asin(*ausdruck*) | berechnet den Arcussinus |
| acos(*ausdruck*) | berechnet den Arcuscosinus |
| atan(*ausdruck*) | berechnet den Arcustangens |
| log(*ausdruck*) | berechnet den natürlichen Logarithmus |
| log10(*ausdruck*) | berechnet den 10-er Logarithmus |
| sqrt(*ausdruck*) | berechnet die Quadratwurzel |
| exp(*ausdruck*) | berechnet den Exponentialwert $e^x$ |
| pow(x,y) | berechnet den Exponentialwert $x^y$ |

***Strukturbefehle:***

| | |
|---|---|
| if *(bedingung*)<br>{<br>*anweisungen;*<br>...<br>}<br>else<br>{<br>*anweisungen;*<br>...<br>} | Entscheidungen |
| while(*bedingung*)<br>{<br>*anweisungen;*<br>...<br>} | kopfgesteuerte Schleife |
| do<br>{<br>*anweisungen;*<br>...<br>}<br>while(*bedingung*); | fußgesteuerte Schleife |
| for(i=0;i<10;i++)<br>{<br>*anweisungen;*<br>…<br>} | zählergesteuerte Schleife<br>(Beispiel) |
| break; | Schleifenabbruch |
| switch(*ausdruck*)<br>{<br>case *konstante:*<br>*anweisungen*;<br>...; break;<br>case *konstante:*<br>*anweisungen*;<br>...; break;<br>...<br>default:<br>*anweisungen*;<br>} | Fallunterscheidung |
| return(*ausdruck*) | Rücksprung aus Funktion |

***Bedingungoperatoren:***

| | |
|---|---|
| == | gleich |
| != | ungleich |
| < | kleiner als |
| > | größer als |
| <= | kleiner oder gleich |
| >= | größer oder gleich |
| && | logisches UND |
| \|\| | logisches ODER |

***Ein-/Ausgabefunktionen:***

***Konsolenausgabe:***

printf("...*formatstring*...",*ausdruck1*, *ausdruck2*, ...)

***Konsoleneingabe:***

scanf("*format*",*&bezeichner*)

***Steuer- und Formatierungszeichen:***

| | |
|---|---|
| \n | Neue Zeile |
| \r | Zeilenrücklauf |
| \t | Tabulator |
| \a | Alarmton |
| %i | Platzhalter ganze Zahlen |
| %f | Platzhalter Fließkommazahlen(float) |
| %lf | Platzhalter Fließkommazahlen(double) |
| %u | Platzhalter große ganze Zahlen |
| %c | Platzhalter einzelnes Zeichen |
| %x | Platzhalter Hex-Zahl |
| %s | Platzhalter String |

***Beispiele:***

| | |
|---|---|
| ...%3i... | 3-stellige ganze Zahl |
| ...%8.2f... | 8-stellige float-Zahl (inkl. '.') mit zwei Nachkommastellen |
| ...%20s... | 20-stelliger String |
| scanf("%f",&zahl); | Eingabe float-Zahl |

***Tastatureingabe:***

| | |
|---|---|
| getch() | warte auf Tastenbetätigung<br>Rückgabewert = ASCII-Zeichen |
| getche(); | wie getch() mit Zeichenausgabe auf Konsole |

***Zufallszahlen:***

srand((unsigned)time(NULL)); Initialisierung

| | |
|---|---|
| rand(); | Rückgabewert: Zufallszahl von 0…32767 |

***Beispiel:***

| | |
|---|---|
| z = (rand()%10)+1; | z ist eine ganzzahlige Zufallszahl von 1 bis 10 |

***Stringoperationen:*** *text*, *text1*, *text2* sind Zeiger auf Strings

***Deklarationsbeispiele:***

| | |
|---|---|
| char text[20]; | 20-stelliger String<br>(inkl. abschließender 0) |
| char name[10][30]; | mehrdimensionales Datenfeld<br>10 Namen mit je 30 Zeichen |

***Funktionen:***

| | |
|---|---|
| gets_s(*text*); | Stringeingabe |
| strlen(*text*); | Stringlänge (ohne abschließende 0) ermitteln |
| strcpy_s(*text1*,*text2*); | text2 nach text1 kopieren |
| strcat_s(*text1*,*text2*); | text2 an text1 anhängen |
| strcmp(*text1*,*text2*); | text1 mit text2 vergleichen<br>Rückgabewert:<br>0, wenn text1 gleich text2<br><0, wenn text1 kleiner als text2<br>>0, wenn text1 größer als text2 |

***Sonstige Funktionen:***

| | |
|---|---|
| system("cls"); | Konsolenbildschirm löschen |
| toupper(*c*); | Zeichen nach Großbuchstaben wandeln |

***Strukturen:***

```
struct structname
{
typ elementname;
typ elementname:
… ;
};              // ';' erforderlich!
```

***Zugriff auf Elemente:***

| | |
|---|---|
| structname.elementname | ‚.'-Operator |
| structnamezeiger->elementname | ‚->'-Operator |

***Klassen:***

```
Class CName
{
private|protected|public:
typ element;
typ element;
…;
private|protected|public:
CName();                    //Standardkonstruktor
CName(typ, typ, …);         //allgemeiner Konstruktor
        //allgemeiner Konstruktor mit Vorgabewerten:
CName(typ bezeichner=wert, typ bezeichner=wert, …);
typ methode();
typ methode();
friend typ friendmethode(); //friend-Methoden
…;
};
```

***Initialisierung der Konstruktoren und Methoden:***

```
CName::CName[(…)]       //Initialisierung Konstruktor
{
element=wert;
…
}

typ CName::methode()    //Initialisierung Methoden
{
anweisungen;
…;
}
```

***Vererbung:***

```
CAname:public CName     //CName = Basisklasse
{
private|protected|public:
typ element;
…;
private|protected|public:
CAname();                //Standardkonstruktor
CAname(typ, typ, …);     //allgemeiner Konstruktor
typ methode();
…;
}
```

***Dateioperationen:***

***Datei öffnen:***

FILE* fopen( const char* filename, const char* mode );

FILE* Zeiger auf Datei; der Rückgabewert ist NULL, wenn das Öffnen fehlschlägt

| | |
|---|---|
| filename | Pfad und Dateiname |
| mode | Zugriffsmode |

***Tabelle für den Zugriffsmode:***

| | |
|---|---|
| "r" | Öffnet Datei zum Lesen. Wenn die Datei nicht existiert liefert fopen den Wert NULL |
| "w" | Öffnet Datei zum Schreiben. Wenn die Datei nicht existiert wird sie angelegt. Eine vorhandene Datei wird überschrieben. |
| "a" | Öffnet Datei zum Schreiben. Die geschriebenen Daten werden an die Datei angehängt (append). Wenn die Datei nicht existiert wird sie neu angelegt. |
| "r+" | Öffnet Datei zum Lesen und Schreiben. Wenn die Datei nicht existiert liefert fopen den Wert NULL |
| "w+" | Öffnet Datei zum Lesen und Schreiben. Wenn die Datei nicht existiert wird sie angelegt. |
| "a+" | Öffnet Datei zum Lesen und Schreiben. Die geschriebenen Daten werden an die Datei angehängt (append). Wenn die Datei nicht existiert wird sie neu angelegt. |
| "…t" | Öffnen als Textdatei, d.h. LF-Zeichen werden mit CR-Zeichen ergänzt |
| „…b" | Öffnen als binäre Datendatei |

***Datei schließen:***

fclose(FILE* stream); stream = Zeiger auf Datei

***Dateiende abfragen:***

int feof( FILE* stream ); Der Rückgabewert ist 0, wenn das Dateiende erreicht ist

***Daten in Datei schreiben:***

fprintf( FILE* stream, const char* format [, argument ]... );

| | |
|---|---|
| stream | Zeiger auf Datei |
| format | Formatangabe; wie printf()-Funktion |
| argument | Ausgabewerte |

***Textdaten lesen:***

char* fgets( char* str, int n, FILE* stream );

| | |
|---|---|
| str | Zeiger auf Zieldaten |
| n | maximale Anzahl zu lesender Zeichen |
| stream | Zeiger auf Datei |

***Einzelnes Zeichen lesen:***

int fgetc( FILE* stream ); Rückgabewert ist der ASCII-Code des gelesenen Zeichens

***Binäre Daten schreiben:***

int fwrite(char* buffer, int size, int count, FILE* stream );

| | |
|---|---|
| buffer | Zeiger auf die zu speichernden Daten |
| size | Länge eines Datenfeldes in Byte |
| count | Anzahl der zu speichernden Datenfelder |
| stream | Zeiger auf Datei in der gespeichert wird |

Der Rückgabewert ist die Anzahl der gespeicherten Datenfelder.

***Binäre Daten lesen:***

int fread(char* buffer, int size, int count, FILE* stream );

| | |
|---|---|
| buffer | Zeiger auf Speicherstelle (Ziel) |
| size | Länge eines Datenfeldes in Byte |
| count | Max. Anzahl der zu lesenden Datenfelder |
| stream | Zeiger auf Datei aus der gespeichert wird |

Der Rückgabewert ist die Anzahl der tatsächlich gelesenen Datenfelder.