


```

    else {
        if (p != s) {
            *p = '\0';
            return s;
        }
    }
}
return NULL;
}

char *getword_utf8() {
    static int n=0; // static bedeutet, dass die Variablen nach
    static char s[200]; // Beendigung der Funktion bestehen bleiben
    static wchar_t wline[2000], *wlp=wline;
    // auf schon angefangener Zeile Wortanfang suchen
    if (n>0) {
        while (*wlp && !iswalp(*wlp)) wlp++; // finde Wortanfang
        if (!*wlp) n=0; // Zeile zu Ende
    }
    // falls Zeile zu Ende: neue Zeile lesen und Wortanfang suchen
    while (n==0) {
        string line;
        if (!getline(cin,line)) // Zeile lesen
            return NULL; // falls EOF return NULL
        n = mbstombs(wline,line.c_str(),2000); // Zeile konvertieren
        wlp = wline;
        while (*wlp && !iswalp(*wlp)) wlp++; // finde Wortanfang
        if (!*wlp) n=0; // Zeile zu Ende
    }
    // nun haben wir einen Wortanfang
    wchar_t w[200], *wp = w;
    while (iswalp(*wlp)) *wp++=*wlp++; // kopiere Wort
    *wp = L'\0'; // Ende markieren
    wcstombs(s,w,200); // nach UTF-8 konvertieren
    return s; // zurueckgeben
}

int main() {
    tree<string> tr; // Deklariere Baum aus Strings
    setlocale(LC_ALL, "de_DE.UTF-8"); // cf. "man setlocale", "man locale"
    while ( char *p = getword_utf8() ) {
        tr.insert(p);
    }
    inorder(cout, tr); // Ausgabe der sortierten Woerter
}

```

Die Klasse `tree` ist hier als Template mit "generischem" Datentyp `T` definiert und hat nur eine Komponente, einen Zeiger auf eine Struktur `node`. Diese Struktur nun trägt hier die oben genannten Komponenten und besitzt selbst eine Konstruktorfunktion.

Es gibt zwei Konstruktoren für `tree`, eine Member-Funktion `insert()` für das Einsetzen eines neuen Wortes, und eine Friend-Funktion `inorder()` zur (geordneten) Ausgabe.

In der Funktion `insert()` wird per `new node` Speicherplatz für ein Objekt vom Typ `node` allokiert. `new` ist C++-Schlüsselwort und liefert, aufgerufen für einen Datentyp, einen Zeiger auf diesen Typ, der auf freien Speicherplatz für "eine Instanz" dieses Typs zeigt. Hat der Typ einen Konstruktor ohne Argument, wird

dieser automatisch dabei aufgerufen. `new` erfüllt also eine ähnliche Funktion in C++ wie `malloc()` für C (cf. 'man malloc'), aber initialisiert den Speicher darüberhinaus nach Maßgabe eines Konstruktors. (Mit `delete`, angewandt auf durch `new` gewonnenen Speicher, kann man diesen wieder freigeben.)

Im weiteren Code bedeutet die Anweisung

```
n->l = tree(n->l).insert(e);
```

folgendes: Für den linken Teilbaum `tree(n->l)` wird die Member-Funktion `insert()` mit dem Argument `e` aufgerufen, Resultat ist ein neuer `node` (mit dem alten linken Teilbaum, um einen `node` mit `e` erweitert), dieser wird `n`-Komponente des neuen linken Teilbaums. (Genaugenommen wird hier nur dann eine echte Zuweisung vorgenommen, wenn der linke Teilbaum vorher leer war, also `n->l == NULL`.)

Schließlich wird der `node this->n` zurückgegeben, `this` (als Schlüsselwort in C++) bedeutet in einer Member-Funktion einen Zeiger auf das implizite Argument, also auf das Objekt selbst, zu dem die Member-Funktion gehört. `*this` ist also das Objekt – in diesem Fall der linke Teilbaum – selbst, mit der `n`-Komponente `this->n`.

Der Aufruf für das Programm kann – wenn der Name der kompilierten Datei `btree` ist – zum Beispiel geschehen durch

```
./btree < textdatei | less
```

wobei die Eingabe dann aus einer (auch sehr grossen) Textdatei genommen wird, deren einzelne Wörter dann alphabetisch mit Häufigkeitsangabe ausgegeben wird.

Aufgabe 9.1: Studieren Sie die Funktion des Unterprogramms `getword` in allen Einzelheiten, inclusive `man isalpha`. Wie werden hier Satzzeichen oder "White Space" aus der Textdatei behandelt?

Aufgabe 9.2: Erweitern Sie die Funktionalität, so dass zu jedem Knoten=Wort auch noch die Zeilenzahlen seines Auftretens festgehalten werden.

Hier könnte eine weitere Strukturkomponente in der Struktur `struct node` hilfreich sein: Ein weiterer `String`, der die Zeilennummern aufnimmt.

Das Programm verwendet den Datentyp `string` aus der C++ Standardbibliothek, siehe hierzu

<http://www.cplusplus.com/reference/string/>

Kommandozeilen-Argumente werden in C++ wie in C behandelt.

Der Zugriff auf Dateien ist einfach: Durch den Befehl `ifstream from(argv[1]);` wird ein Objekt `from` vom Typ `ifstream` (Eingabestrom) definiert, über den die Zeichen der Datei `argv[1]` mit den üblichen Mitteln wie für `cin` (`cin.get(),...`) der Reihe nach gelesen werden können. Dabei ist `from` ein frei gewählter Name. Die Anweisung `ifstream from(argv[1]);` ist analog zu einer Deklaration mit Initialisierung wie etwa `int anzahl = 3;` zu sehen: `ifstream` ist ein Datentyp, `from` das Objekt mit frei gewähltem Namen (wie `anzahl`, das initialisiert wird auf die Datei namens `argv[1]`). Ganz analog arbeitet der Befehl `ofstream to(argv[2]);`; hier ist `to` das Objekt vom Typ `ofstream`, eine Datei, in die geschrieben wird.

```
// fileio.cc  Lesen und Schreiben von Dateien

using namespace std;
#include <iostream>
#include <fstream>
#include <stdlib.h>

void error(int i, char* s, string t = "") {
    cerr << s << ' ' << t << '\n'; exit(i);
}

int main(int argc, char* argv[] ) {

    if (argc != 3) {
        cerr << "Falscher Aufruf, korrekter Aufruf:\n";
        error (1, argv[0], "Quelldatei Zieldatei");
    }

    ifstream from(argv[1]); // Deklaration eines ifstream namens from
                           // der aus der Datei namens argv[1] liest.
    if (!from) error(2, argv[1], "kann nicht zum Lesen geoeffnet werden.\n");

    ofstream to(argv[2]);
    if (!to) error(3, argv[2], "kann nicht zum Schreiben geoeffnet werden.\n");

    char c;

    while (from.get(c))
        to.put(c);

    from.close(); to.close();

    return 0;
}
```

Aufgabe 9.3: Testen Sie das kompilierte Programm `fileio`, indem Sie Befehle wie

```
fileio quelle ziel; echo $?
```

eingeben, wobei "quelle" und "ziel" auch weggelassen werden bzw. durch Namen nicht existenter Dateien ersetzt werden. Die Status-Variable `$?` zeigt dann den durch `exit` zurückgegebenen `int`-Wert. Die Status-Variable `$?` gilt für die `bash`-Shell. Für `csh` oder `tcsh` lautet sie `$status` .

Aufgabe 9.4: Modifizieren Sie das Programm `fileio.cc` so, dass ein Aufruf des ausführbaren Programms `fileio` folgendes tut:

```
./fileio (ohne Parameter): Programm liest von der Tastatur (cin) und schreibt die gelesenen Daten auf den Bildschirm (nach cout).
./fileio quelle (ein Parameter): Liest aus Datei quelle, schreibt den Inhalt nach cout,
./fileio quelle ziel (zwei Parameter): Liest aus Datei quelle, schreibt den Inhalt in die Datei ziel,
./fileio quelle_1 ... quelle_n ziel (mehr als zwei Parameter): Liest der Reihe nach aus den Dateien
quelle_1, ..., quelle_n und schreibt alles in die Datei ziel, die Dateien quelle_1, ... quelle_n
werden also aneinanderghängt.
```

Aufgabe 9.5: Benutzen Sie die Programme `fileio.cc` und `btree()`, um ein Programm zu schreiben, das die Wörter aus einer Textdatei liest und alphabetisch geordnet mit Häufigkeitsangabe ausgibt. (Hinweis: Benutzen Sie z. B. das Programm `cc.cc` von Blatt 6, um Wörter zu erkennen.)