```c
#include <stdio.h>
#define LETTER 'a'
#define DIGIT '0'
#define BUFSIZE 100
#define MAXWORD 20
char buf[BUFSIZE];
int bufp = 0;

struct tnode {
        char *word;
        int count;
        struct tnode *left;
        struct tnode *right;
};

struct tnode *tree();
char *strsave();
struct tnode *talloc();

main() {
        struct tnode *root, *tree();
        char word[MAXWORD];
        int t;

        root = NULL;
        while ((t = getword(word,MAXWORD)) != EOF)
                if ( t == LETTER) root = tree(root,word);
        treeprint(root);
}


getword(w,lim)  /* 137p */
char *w; int lim;
{
        int c,t;
        if (type(c = *w++ = getch()) != LETTER) {
                *w = '\0';
                return(c);
        }
        while (--lim>0) {
                t = type(c = *w++ = getch());
                if (t!=LETTER && t != DIGIT) {
                        ungetch(c);
                        break;
                }
        }
        *(w-1) = '\0';
        return(LETTER);
}

type(c) /* 138p */
int c;
{
        if ( c >= 'a' && c<= 'z' || c >='A' && c <='Z') return(LETTER);
        else if ( c >= '0' && c <= '9') return(DIGIT);
        else return(c);
}

getch() /* 87p */
{
        return((bufp > 0) ? buf[--bufp] : getchar());
}

ungetch(c)
int c;
{
```

printf("sizeof(struct tnode*)=%d\n", sizeof(struct tnode *));


```c
        if (bufp > BUFSIZE) printf("ungetch: too many characters\n");
        else buf[bufp++] = c;
}


struct tnode *tree(p,w) /* 143p */
struct tnode *p;
char *w;
{
        struct tnode *talloc();
        char *strsave();
        int cond;

        if ( p == NULL) {
                p = talloc();
                p->word = strsave(w);
                p->count = 1;
                p->left=p->right=NULL;
        }else if ((cond = strcmp(w,p->word)) == 0) p->count++;
        else if (cond < 0) p->left = tree(p->left,w);
        else p->right = tree(p->right,w);
        return(p);
}

char *strsave(s) /* 112p */
char *s;
{
        char *p,*malloc();

        if ((p = malloc(strlen(s)+1)) != NULL) strcpy(p,s);
        return(p);
}

treeprint(p) /* 144p */
struct tnode *p;
{
        if ( p != NULL) {
                treeprint(p->left);
                printf("%4d %s\n",p->count,p->word);
                treeprint(p->right);
        }
}

struct tnode *talloc() /* 145p */
{
        char *malloc();
        return((struct tnode *) malloc(sizeof(struct tnode)));
}
```

printf("%p %s\n", tree, word);

出力されたアドレスから、メモリの図を書く。

printf("%p\n", p);