тобы завершить обсуждение двусвязных списков, в данном разделе представлена простая, но законченная программа для работы со списком рассылки. Во время работы весь список хранится в оперативной памяти. Тем не менее, его можно сохранять в файле и загружать для дальнейшей работы.

/\* Простая программа для обработки списка рассылки

иллюстрирующая работу с двусвязными списками.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct address {

char name[30];

char street[40];

char city[20];

char state[3];

char zip[11];

struct address \*next; /\* указатель на следующую запись \*/

struct address \*prior; /\* указатель на предыдущую запись \*/

};

struct address \*start; /\* указатель на первую запись списка \*/

struct address \*last; /\* указатель на последнюю запись \*/

struct address \*find(char \*);

void enter(void), search(void), save(void);

void load(void), list(void);

void mldelete(struct address \*\*, struct address \*\*);

void dls\_store(struct address \*i, struct address \*\*start,

struct address \*\*last);

void inputs(char \*, char \*, int), display(struct address \*);

int menu\_select(void);

int main(void)

{

start = last = NULL; /\* инициализация указателей на начало и конец \*/

for(;;) {

switch(menu\_select()) {

case 1: enter(); /\* ввод адреса \*/

break;

case 2: mldelete(&start, &last); /\* удаление адреса \*/

break;

case 3: list(); /\* отображение списка \*/

break;

case 4: search(); /\* поиск адреса \*/

break;

case 5: save(); /\* запись списка в файл \*/

break;

case 6: load(); /\* считывание с диска \*/

break;

case 7: exit(0);

}

}

return 0;

}

/\* Выбор действия пользователя. \*/

int menu\_select(void)

{

char s[80];

int c;

printf("1. Ввод имени\n");

printf("2. Удаление имени\n");

printf("3. Отображение содержимого списка\n");

printf("4. Поиск\n");

printf("5. Сохранить в файл\n");

printf("6. Загрузить из файла\n");

printf("7. Выход\n");

do {

printf("\nВаш выбор: ");

gets(s);

c = atoi(s);

} while(c<0 || c>7);

return c;

}

/\* Ввод имени и адресов. \*/

void enter(void)

{

struct address \*info;

for(;;) {

info = (struct address \*)malloc(sizeof(struct address));

if(!info) {

printf("\nНет свободной памяти");

return;

}

inputs("Введите имя: ", info->name, 30);

if(!info->name[0]) break; /\* завершить ввод \*/

inputs("Введите улицу: ", info->street, 40);

inputs("Введите город: ", info->city, 20);

inputs("Введите штат: ", info->state, 3);

inputs("Введите почтовый индекс: ", info->zip, 10);

dls\_store(info, &start, &last);

} /\* цикл ввода \*/

}

/\* Следующая функция вводит с клавиатуры строку

длинной не больше count и предотвращает переполнение

строки. Кроме того, она выводит на экран подсказку. \*/

void inputs(char \*prompt, char \*s, int count)

{

char p[255];

do {

printf(prompt);

fgets(p, 254, stdin);

if(strlen(p) > count) printf("\nСлишком длинная строка\n");

} while(strlen(p) > count);

p[strlen(p)-1] = 0; /\* удалить символ перевода строки \*/

strcpy(s, p);

}

/\* Создание упорядоченного двусвязного списка. \*/

void dls\_store(

struct address \*i, /\* новый элемент \*/

struct address \*\*start, /\* первый элемент списка \*/

struct address \*\*last /\* последний элемент списка \*/

)

{

struct address \*old, \*p;

if(\*last==NULL) { /\* первый элемент списка \*/

i->next = NULL;

i->prior = NULL;

\*last = i;

\*start = i;

return;

}

p = \*start; /\* начать с начала списка \*/

old = NULL;

while(p) {

if(strcmp(p->name, i->name)<0){

old = p;

p = p->next;

}

else {

if(p->prior) {

p->prior->next = i;

i->next = p;

i->prior = p->prior;

p->prior = i;

return;

}

i->next = p; /\* новый первый элемент \*/

i->prior = NULL;

p->prior = i;

\*start = i;

return;

}

}

old->next = i; /\* вставка в конец \*/

i->next = NULL;

i->prior = old;

\*last = i;

}

/\* Удаление элемента из списка. \*/

void mldelete(struct address \*\*start, struct address \*\*last)

{

struct address \*info;

char s[80];

inputs("Введите имя: ", s, 30);

info = find(s);

if(info) {

if(\*start==info) {

\*start=info->next;

if(\*start) (\*start)->prior = NULL;

else \*last = NULL;

}

else {

info->prior->next = info->next;

if(info!=\*last)

info->next->prior = info->prior;

else

\*last = info->prior;

}

free(info); /\* освободить память \*/

}

}

/\* Поиск адреса. \*/

struct address \*find( char \*name)

{

struct address \*info;

info = start;

while(info) {

if(!strcmp(name, info->name)) return info;

info = info->next; /\* перейти к следующему адресу \*/

}

printf("Имя не найдено.\n");

return NULL; /\* нет подходящего элемента \*/

}

/\* Отобразить на экране весь список. \*/

void list(void)

{

struct address \*info;

info = start;

while(info) {

display(info);

info = info->next; /\* перейти к следующему адресу \*/

}

printf("\n\n");

}

/\* Данная функция выполняет собственно вывод на экран

всех полей записи, содержащей адрес. \*/

void display(struct address \*info)

{

printf("%s\n", info->name);

printf("%s\n", info->street);

printf("%s\n", info->city);

printf("%s\n", info->state);

printf("%s\n", info->zip);

printf("\n\n");

}

/\* Поиск имени в списке. \*/

void search(void)

{

char name[40];

struct address \*info;

printf("Введите имя: ");

gets(name);

info = find(name);

if(!info) printf("Не найдено\n");

else display(info);

}

/\* Сохранить список в дисковом файле. \*/

void save(void)

{

struct address \*info;

FILE \*fp;

fp = fopen("mlist", "wb");

if(!fp) {

printf("Невозможно открыть файл.\n");

exit(1);

}

printf("\nСохранение в файл\n");

info = start;

while(info) {

fwrite(info, sizeof(struct address), 1, fp);

info = info->next; /\* перейти к следующему адресу \*/

}

fclose(fp);

}

/\* Загрузка адресов из файла. \*/

void load()

{

struct address \*info;

FILE \*fp;

fp = fopen("mlist", "rb");

if(!fp) {

printf("Невозможно открыть файл.\n");

exit(1);

}

/\* освободить память, если в памяти уже есть список \*/

while(start) {

info = start->next;

free(info);

start = info;

}

/\* сбросить указатели на начало и конец \*/

start = last = NULL;

printf("\nЗагрузка из файла\n");

while(!feof(fp)) {

info = (struct address \*) malloc(sizeof(struct address));

if(!info) {

printf("Нет свободной памяти");

return;

}

if(1 != fread(info, sizeof(struct address), 1, fp)) break;

dls\_store(info, &start, &last);

}

fclose(fp);

}