acm.mipt.ru

олимпиады по программированию на Физтехе

Раздел «Язык Си» . SimpleList :

Реализация односвязного списка

Ниже приведён код, реализующий односвязный список.

Каждый элемент списка (list item) хранит пару (key, value) и указатель на следующий элемент списка.

Если следующего элемента нет, то указатель равен значению NULL.

Ниже показан пример работы с программой. Зелёным цветов выделен текст, который выводит программа, а чёрным – текст, который вводит пользователь.

```
ADD 1 1
ADD 2 2
ADD 3 3
PRINT
Size=3
(3,3) -> (2,2) -> (1,1) -> NULL
DEL 3
PRINT
Size=2
(2,2) -> (1,1) -> NULL
ADD 10 10
ADD 11 11
ADD 0 0
FIND 2
Value = 2
PRINT
Size=5
(0,0) -> (11,11) -> (10,10) -> (2,2) -> (1,1) -> NULL
ADD 2 222
PRINT
Size=5
(0,0) -> (11,11) -> (10,10) -> (2,222) -> (1,1) -> NULL
```

```c
/*
Usage:
  ADD
  DEL
  FIND
  PRINT
*/

#include <stdio.h>
#include <string.h>
#include <malloc.h>

typedef int mkey_t;  // Тип ключа
typedef int value_t; // Тип значения

struct li {
   mkey_t key;
   value_t value;
   struct li* next;
};

struct list {
   struct li *first;
   int size;
};

struct list *new_list(void) {
   struct list *res = (struct list*) malloc(sizeof(struct list));
   res->first = NULL;
   res->size = 0;
```

Поиск
Поиск
Раздел «Язык Си»
Главная
Зачем учить С?
Определения
Инструменты:
Поиск
Изменения
Index
Статистика
Разделы
Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C
Logon>>

```
    return res;
}

struct li* new_list_item(void) {
     struct li *res = (struct li*) malloc(sizeof(struct li));
     res->next = 0;
     return res;
}

void delete_items(struct li* it) {
   if ( it ) {
      if ( it->next ) {
         delete_items(it->next);
      }
      free( it );
   }
}

void delete_list(struct list *l) {
   if ( l ) {
      delete_items( l->first );
      free ( l );
   }
}

/* Добавляет элемент (key, value) в начало списка l
 * Возвращает 1, если удалось добавить элемент (память под элемент была успешно выделена)
 */
int insert_item( struct list *l, mkey_t key, value_t value ) {
   struct li *it = new_list_item();
   if ( it == NULL ) return 0;
   it->key = key;
   it->value = value;
   it->next = l->first;
   l->first = it;
   l->size++;
   return 1;
}

/* Добавляет элементы (key, value) в начало списка l только в том случае,
 * если список l не содержит элемента с ключём key.
 * Возвращает 1, если удалось добавить элемент (память под элемент была успешно выделена)
 * или обновить значение value.
 */
int insert_item_uniq( struct list *l, mkey_t key, value_t value ) {
   if ( l ) {
      struct li *it;
      for ( it = l->first; it != NULL; it = it->next ) {
         if (it->key == key) {
            it->value = value;
            return 1;
         }
      }
      return insert_item(l, key, value);

   } else {
      return 0;
   }
}

/* Удаляет элемент (key, *).
 * Возвращает 1, если элемент с указанным ключём был найден в списке l.
 */
int delete_item( struct list *l, mkey_t key ) {
   struct li *it, *prev_it = 0;
   for (  it = l->first; it != NULL; prev_it = it, it = it->next ) {
      if ( it->key == key ) {
         if ( prev_it != 0 )
            prev_it->next = it->next;
         else
            l->first = it->next;
         free( it );
```

```
            l->size--;
            return 1;
         }
      }
      return 0;
}

/* Второй вариант реализации удаления элемента из списка
*/
/*
int delete_item2( struct list *l, mkey_t key ) {
   struct li **it = &(l->first);
   for ( ; (*it) != NULL; it = &( (*it)->next) ) {
      if ( (*it)->key == key ) {
         struct li *tmp = *it;
         *it = (*it)->next ;
         free( tmp );
         return 1;
      }
   }
   return 0;
}
*/

/*
 *
 */
void print_list ( struct list *l ) {
   struct li *it;
   printf( "Size=%d\n", l->size );
   for ( it = l->first; it != NULL; it = it->next ) {
      printf("(%d,%d) -> ", it->key, it->value);
   }
   printf( "NULL\n" );
}

int find_item( struct list *l, mkey_t key, value_t *value) {
   struct li *it;
   for ( it = l->first; it != NULL; it = it->next ) {
      if ( it->key == key ) {
         *value = it->value;
         return 1;
      }
   }
   return 0;
}


int main()
{
   char cmd[1024];
   struct list *l = new_list();
   while (1) {
      fgets( cmd, sizeof(cmd), stdin );
      if ( strncmp( cmd, "ADD", 3 ) == 0 ) {
         mkey_t key;
         value_t value;
         if( sscanf(cmd + 3, "%d%d", &key, &value) == 2 ) {
            insert_item_uniq(l, key, value );
         } else {
            printf("Bad arguments: %s\n", cmd+3);
            printf("Usage: ADD <key> <value>\n");
         }
      } else if ( strncmp( cmd, "DEL", 3 ) == 0 ) {
         mkey_t key;
         sscanf(cmd + 3, "%d", &key );
         delete_item( l, key );
      } else if ( strncmp( cmd, "PRINT", 5) == 0 ) {
         print_list( l );
      } else if ( strncmp( cmd, "FIND", 4) == 0 ) {
         mkey_t key;
         value_t value;
```

```
            sscanf( cmd + 4, "%d", &key );
            if ( find_item( l, key, &value ) ) {
               printf("Value = %d\n", value);
            } else {
               printf("Not found\n");
            }
         }
      }

      return 0;
}
```

Задача. Модифицируйте данный код так, чтобы он решал задачу "Телефонная книжка", а именно,сделайте тип ключа равным `char*` и внесите изменения в строчки сравнения ключей, а также в функцию `new_list_item` создания нового элемента списка. Добавьте к функции `new_list_item` аргументы `key`, `value`.