**Міністерство освіти та науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут»**

**Факультет прикладної математики**

**Кафедра системного програмування і спеціалізованих**

**комп’ютерних систем**

**Лабораторна робота №2**

з дисципліни

**«Об’єктно-орієнтоване програмування»**

Тема: **«Структури»**

Виконав: Місік Дмитро Сергійович

Студент групи КВ-31

Перевірив(ла):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ 2014

1. **Файл listtest.c**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

name: listtest.c

description: this file call functions, that described in file "list.c" to

test performance of this functions

author: Dima

date of creation: 07.09.2014

written: 07.09.2014

date of last modified: 10.09.2014

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "list.h"

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

int main**()** **{**

CNode **\***head1 **=** **NULL,** **\***head2 **=** **NULL,** **\***head **=** **NULL;**

int i**;**

**for** **(**i **=** 0**;** i **<** 10**;** i**++)** **{**

CNode **\***p **=** **(**CNode **\*)**malloc**(sizeof(**CNode**));**

p**->**id **=** i**;**

append2list**(&**head1**,** p**);**

**}**

**for** **(**i **=** 5**;** i **<** 15**;** i**++)** **{**

CNode **\***p **=** **(**CNode **\*)**malloc**(sizeof(**CNode**));**

p**->**id **=** i**;**

append2list**(&**head2**,** p**);**

**}**

head **=** merge\_unique**(**head1**,** head2**);**

printf**(**"Firstly\nList 1 = "**);**

print\_list**(**head1**);**

printf**(**"List 2 = "**);**

print\_list**(**head2**);**

printf**(**"List unique = "**);**

print\_list**(**head**);**

printf**(**"\n"**);**

del\_node**(&**head1**,** 9**);**

del\_node**(&**head2**,** 5**);**

del\_node**(&**head**,** 7**);**

printf**(**"After delete\nList 1 = "**);**

print\_list**(**head1**);**

printf**(**"List 2 = "**);**

print\_list**(**head2**);**

printf**(**"List unique = "**);**

print\_list**(**head**);**

printf**(**"\n"**);**

**{**

CNode **\***p **=** **(**CNode **\*)**malloc**(sizeof(**CNode**));**

p**->**id **=** 9**;**

ins\_node**(&**head1**,** p**,** 0**);**

p**->**id **=** 6**;**

ins\_node**(&**head2**,** p**,** 6**);**

p**->**id **=** 9**;**

ins\_node**(&**head**,** p**,** 7**);**

**}**

printf**(**"After insert\nList 1 = "**);**

print\_list**(**head1**);**

printf**(**"List 2 = "**);**

print\_list**(**head2**);**

printf**(**"List unique = "**);**

print\_list**(**head**);**

printf**(**"\n"**);**

reverse**(**head1**);**

reverse**(**head2**);**

reverse**(**head**);**

printf**(**"After reversing\nList 1 = "**);**

print\_list**(**head1**);**

printf**(**"List 2 = "**);**

print\_list**(**head2**);**

printf**(**"List unique = "**);**

print\_list**(**head**);**

printf**(**"\n"**);**

unique**(&**head1**);**

unique**(&**head2**);**

unique**(&**head**);**

printf**(**"After unique\nList 1 = "**);**

print\_list**(**head1**);**

printf**(**"List 2 = "**);**

print\_list**(**head2**);**

printf**(**"List unique = "**);**

print\_list**(**head**);**

printf**(**"\n"**);**

clear**(&**head1**);**

clear**(&**head2**);**

clear**(&**head**);**

**if** **(**isempty**(**head1**))** printf**(**"List 1 is empty\n"**);**

**if** **(**isempty**(**head2**))** printf**(**"List 2 is empty\n"**);**

**if** **(**isempty**(**head**))** printf**(**"List unique is empty\n"**);**

\_getch**();**

**return** 0**;**

**}**

1. **Файл list.c**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

name: list.c

description: file contains functions "list\_last" that returns pointer to

last node in list or NULL if list is empty.

function "append2list" add node to list. Node - copy of

parameter pn. Also, you can add new list in the end of

the list. In copy changes only pointer to previous node.

parameter head - head of list. is list is empty - function

create list, changes parameter head. else - add node in list

which starts from head

function "create\_list" returns pointer to head to list

this function creates list uses function "append2list"

function "isempty" return 0, if list has nodes and returns

1 if list empty (list empty if head == NULL)

function "clear" clear list, which starts from head

function "get\_node" returns pointer to first node, that

id is id or NULL if list doesn't contain node with id id

function "del\_node" deletes all nodes that id is id and change

list and head if it is necessary

function "ins\_node" insert copy of pn before ellement with id

id into list which starts from head. Function change in copy

pointer to previous and next nodes

function "reverse" reverse list and change head necessarily

function "print\_list" displays list, if it isn't empty or

print 'Empty list' if list is empty

functon "isunique" check, if nodes p1 id unique in list,

which starts from p2 (node id unique - id doesn't meet in

list p2 or meets once and pointer p1 point the same adress

that pointer in list p2). if id of p1 is unique - function

returns 1, else - 0.

function "merge\_unique" creates list of unique ides, that

meets in list with start in head1 and list that start in

head2. if it is possible - returns pointer to head of unique

list, else returns NULL.

function "unique" deletes in list which

starts in head all neighbours nodes that id is the same.

author: Dima

date of creation: 07.09.2014

written: 07.09.2014

date of last modified: 10.09.2014

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "list.h"

#include <stdlib.h>

#include <stdio.h>

int isempty**(**const CNode **\***head**)** **{**

**if** **(**head **==** **NULL)** **return** 1**;**

**else** **return** 0**;**

**}**

CNode **\***get\_node**(**const CNode **\***head**,** int id**)** **{**

CNode **\***p **=** head**;**

**while** **(**p**->**id **!=** id **&&** p**->**next **!=** head**)** p **=** p**->**next**;**

**if** **(**p**->**id **==** id**)** **return** p**;**

**else** **return** **NULL;**

**}**

void append2list**(**CNode **\*\***head**,** const CNode **\***pn**)** **{**

CNode **\***p **=** **(**CNode **\*)**malloc**(sizeof(**CNode**));**

p**->**id **=** pn**->**id**;**

**if** **(\***head **==** **NULL)** **{**

**\***head **=** p**;**

**}**

**else** **{**

**(\***head**)->**prev**->**next **=** p**;**

p**->**prev **=** **(\***head**)->**prev**;**

**}**

p**->**next **=** **\***head**;**

**(\***head**)->**prev **=** p**;**

**}**

void del\_node**(**CNode **\*\***head**,** int id**)** **{**

CNode **\***p **=** **\***head**;**

**if** **(!**isempty**(\***head**))** **do** **{**

**if** **(**p**->**id **==** id**)** **{**

CNode **\***temp **=** p**;**

p**->**prev**->**next **=** p**->**next**;**

p**->**next**->**prev **=** p**->**prev**;**

**if** **(**p **==** **\***head**)** **\***head **=** p**->**prev**->**next**;**

p **=** p**->**next**;**

free**(**temp**);**

**}**

**else** p **=** p**->**next**;**

**}** **while** **(**p **!=** **\***head**);**

**}**

void clear**(**CNode **\*\***head**)** **{**

CNode **\***p **=** **\***head**;**

p**->**prev**->**next **=** **NULL;**

**while** **(**p **!=** **NULL)** **{**

**\***head **=** **(\***head**)->**next**;**

free**(**p**);**

p **=** **\***head**;**

**}**

**}**

void ins\_node**(**CNode **\*\***head**,** const CNode **\***pn**,** int id**)** **{**

CNode **\***p\_id **=** get\_node**(\***head**,** id**);**

**if** **(**p\_id **!=** **NULL)** **{**

CNode **\***p **=** **(**CNode **\*)**malloc**(sizeof(**CNode**));**

p**->**id **=** pn**->**id**;**

p**->**prev **=** p\_id**->**prev**;**

p**->**next **=** p\_id**;**

p\_id**->**prev**->**next **=** p**;**

p\_id**->**prev **=** p**;**

**if** **(**p\_id **==** **\***head**)** **\***head **=** p\_id**->**prev**;**

**}**

**}**

void reverse**(**CNode **\***head**)** **{**

CNode **\***p **=** head**;**

**if** **(!**isempty**(**head**))** **do** **{**

CNode **\***temp **=** p**->**next**;**

p**->**next **=** p**->**prev**;**

p**->**prev **=** temp**;**

p **=** p**->**prev**;**

**}** **while** **(**p **!=** head**);**

**}**

void print\_list**(**const CNode **\***head**)** **{**

CNode **\***p **=** head**;**

**do** **{**

printf**(**"%d, "**,** p**->**id**);**

p **=** p**->**next**;**

**}** **while** **(**p**->**next **!=** head**);**

printf**(**"%d\n"**,** p**->**id**);**

**}**

CNode **\***merge\_unique**(**const CNode **\***head1**,** const CNode **\***head2**)** **{**

CNode **\***head **=** **NULL;**

CNode **\***p **=** head1**;**

**if** **(!**isempty**(**head1**))** **do** **{**

CNode **\***p1 **=** head2**;**

**while** **(**p**->**id **!=** p1**->**id **&&** p1**->**next **!=** head2**)** p1 **=** p1**->**next**;**

**if** **(**p**->**id **!=** p1**->**id**)** append2list**(&**head**,** p**);**

p **=** p**->**next**;**

**}** **while** **(**p **!=** head1**);**

p **=** head2**;**

**if** **(!**isempty**(**head2**))** **do** **{**

CNode **\***p1 **=** head1**;**

**while** **(**p**->**id **!=** p1**->**id **&&** p1**->**next **!=** head1**)** p1 **=** p1**->**next**;**

**if** **(**p**->**id **!=** p1**->**id**)** append2list**(&**head**,** p**);**

p **=** p**->**next**;**

**}** **while** **(**p **!=** head2**);**

**return** head**;**

**}**

void unique**(**CNode **\*\***head**)** **{**

CNode **\***p **=** **\***head**;**

**do** **{**

**if** **(**p**->**id **==** p**->**next**->**id**)** **{**

**if** **(**p **==** **\***head**)** **\***head **=** **(\***head**)->**next**->**next**;**

**else** **if** **(**p**->**next **==** **\***head**)** **\***head **=** **(\***head**)->**next**;**

p**->**prev**->**next **=** p**->**next**->**next**;**

p**->**next**->**next**->**prev **=** p**->**prev**;**

CNode **\***temp **=** p**;**

p **=** p**->**next**->**next**;**

free**(**temp**->**next**);**

free**(**temp**);**

**}**

**else** p **=** p**->**next**;**

**}** **while** **(**p **!=** **\***head**);**

**}**

1. **Файл list.h**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

name: list.h

description: this file contains prototypes, that described in file "list.c"

and description of strcture CNode

author: Dima

date of creation: 07.09.2014

written: 07.09.2014

date of last modified: 10.09.2014

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**typedef** struct tag\_CNode **{**

//attributes

int id**;**//node's inex

struct tag\_CNode **\***prev**;**//Pointer to the previous node in the node's list

struct tag\_CNode **\***next**;**//Pointer to the next node in the node's list

**}** CNode**;**

int isempty**(**const CNode **\***head**);**

CNode **\***get\_node**(**const CNode **\***head**,** int id**);**

void append2list**(**CNode **\*\***head**,** const CNode **\***pn**);**

void del\_node**(**CNode **\*\***head**,** int id**);**

void clear**(**CNode **\*\***head**);**

void ins\_node**(**CNode **\*\***head**,** const CNode **\***pn**,** int id**);**

void reverse**(**CNode **\***head**);**

void print\_list**(**const CNode **\***head**);**

CNode **\***merge\_unique**(**const CNode **\***head1**,** const CNode **\***head2**);**

void unique**(**CNode **\*\***head**);**