Министерство образования и науки РФ

Тамбовский государственный технический университет.

Кафедра:

Отчетпо лабораторной работе №2

***«Динамические структуры данных»***

по дисциплине :

«Программированию на языке высокого уровня»

Вариант №2

Выполнил: студент группы   
.

Проверил:.

Тамбов 20

**Постановка задачи**

1. Написать программу, демонстрирующую использование динамической структуры данных: двунаправленный кольцевой список.
2. Программа должна выполнять следующие действия:

* создание структуры(списка);
* добавление нового элемента структуры (в любое место);
* удаление указанного элемента в списке;
* удаление всего списка;
* поиск информации по ключу.

3. Разработать пользовательский интерфейс.

**Граф-схема**

1-2 – начальное состояние списка и вывод меня выбора на экран.

2-3 – создание списка и добавление нового элемента.

2-4 ­– добавление элемента в начало списка.

2-5 – добавление элемента в конец списка.

2-6а – ввод значения элемента, после которого следует осуществить вставку.

6а-6б – поиск заданного значения элемента.

6б-6в – вставка элемента после найденного.

2-7а – ввод значения элемента, перед которым следует осуществить вставку (исключая первый элемент).

7а-7б – поиск заданного значения элемента.

7б-7в – вставка элемента перед найденным.

2-8а – ввод значения элемента, который следует удалить.

8а-8б – поиск заданного значения элемента.

8б-8в – удаление найденного элемента.

2-9 – удаление всего списка.

2-10 – вывод из меню программы.

10-11 – очистка памяти.

**Исходный текст программы**

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

int main()

{

struct spisok

{

struct spisok \*next, \*prev;

int key;

};

struct spisok \*other, \*first=NULL;

int k, number, choise,n, point=0, message=0;

for (;;)

{ /\*clrscr();\*/

system("cls");

if(message==1)printf("Oshibka\n");

else if (message==2)printf("skisok sozdan\n");

else if (message==3)printf ("takogo elementa net\n");

else if (message==4)printf ("nedostatochno elementov! ( <2 )\n");

printf ("Spisok elementov: ");

message = 0;

if (first==NULL){ printf ("Spisok pust!\n"); }

else

{

other = first;

do

{

printf (" %d",other->key);

other = other->next;

}while (other!=first);

}

printf("\nSdelayte vubor...\n\n");

printf("1-Sozdanie new spiska(dobavlenie 1-go elementa v spisok)\n");

printf("2-Dobavlenie new elementa v nachalo spiska\n");

printf("3-Dobavlenie new elementa v konec spiska\n");

printf("4-Dobavlenie elementa v spisok posle zadanogo elementa\n");

printf("5-Dobavlenie elementa v spisok pered zadannym elementom\n");

printf("6-Udalenie elementa spiska\n");

printf("7-Udalenie spiska\n");

printf("0-Exit\n");

scanf ("%d",&choise);

if (!choise)break;

switch (choise)

{

//---------------------------Sozdanie NEW spiska---------------------------

case 1:{

if (first==NULL)

{

first = (struct spisok\*)malloc(sizeof(struct spisok));

first->next=first->prev=first;

printf ("Nachalnuyu element spiska ");

scanf(" %d", &first->key);

}

else message=2; break;

}

//------------------Dobavlenie new elementa v nachalo spiska---------------------------

case 2:

{

if (first!=NULL)

{

struct spisok \*third;

third = (struct spisok\*)malloc(sizeof(struct spisok));

printf ("Vvedite element v nachalo spiska ");

scanf("%d",&third->key);

third->next=first;

third->prev=first->prev;

first->prev->next=third;

first->prev=third;

first=third;

break;

}

else { message=1; break;}

}

//------------------Dobavlenie new elementa v konec spiska---------------------------

case 3:

if (first!=NULL)

{

struct spisok \*third;

third = (struct spisok\*)malloc(sizeof(struct spisok));

printf ("Vvedite element v konec spiska ");

scanf("%d",&third->key);

third->next=first;

third->prev=first->prev;

first->prev->next = third;

first->prev=third;

break;

}

else { message=1; break;}

//-------------------Dobavlenie elementa v spisok posle zadannogo elementa--------------------------

case 4:

if (first!=NULL&&first->next!=first)

{

struct spisok \*third;

printf ("Posle kakogo elementa v spiske dobavit` new element? ");

scanf ("%d", &number);

other = first;

while (other->key!=number)

{

if (other->next==first){message=3; point =1; break;}

other=other->next;

}

if (point) break;

third=(struct spisok\*)malloc(sizeof(struct spisok));

scanf ("%d",&third->key);

third->prev = other;

third->next=other->next;

other->next->prev = third;

other->next=third;

}

else message =4; break;

//-------------------Dobavlenie elementa v spisok pered zadannym elementom-----------------------------

case 5:

if (first!=NULL&&first->next!=first)

{

struct spisok \*third;

printf ("Pered kakim elementom v spiske dobavit` new element?(ne pered 1!) ");

scanf ("%d", &number);

other = first->next;

while (other->key!=number)

{

if (other->next==first){message=3; point =1; break;}

other=other->next;

}

if (point) break;

third=(struct spisok\*)malloc(sizeof(struct spisok));

scanf ("%d",&third->key);

third->prev = other->prev;

third->next=other;

other->prev->next = third;

other->prev=third;

}

else message =4; break;

//------------------------------Udalenie elementa spiska----------------------------

case 6:

if (first==NULL) message = 1;

else if (first!=NULL&&first->next!=first)

{

printf ("Kakoi element udalit`?");

scanf ("%d", &number);

other = first;

if (other->key==number)

{

first->prev->next=first->next;

first->next->prev=first->prev;

first=first->next;

free (other); point=1;

}

else if (other->prev->key==number)

{

other=first->prev;

first->prev->prev->next=first;

first->prev=first->prev->prev;

free (other); point = 1;

}

if (point) break;

while (other->key!=number)

{

if (other->next==first){message=3; break;}

other=other->next;

}

if (other->key==number)

{

other->prev->next=other->next;

other->next->prev=other->prev;

free(other); break;

}

}

else if (first->next==first)

{

printf ("Kakoi element udalit`? ");

scanf ("%d",&number);

if (first->key==number)

{

free (first);

first=NULL;

}

else message=3;

break;

}

//------------------------------Udalenie spiska----------------------------

case 7:

if (first != NULL)

{

while (first->next!=first)

{

other=first;

first->prev->next=first->next;

first->next->prev = first -> prev;

first=first->next;

free (other);

}

free (first);

first = NULL;

break;

}

}

point = 0;

}

if (first != NULL)

{

while (first->next!=first)

{

other=first;

first->prev->next=first->next;

first->next->prev = first -> prev;

first=first->next;

free (other);

}

free (first);

first = NULL;

}

return 0;

}