Міністерство освіти та науки України
ІФТУНГ

Кафедра Інформаційні
системи та структури даних

Лабораторна робота №5

Структура даних СПИСОК

Виконав студент групи КІ-17-2:
Фрезюк А.І.
Перевірив викладач:
Ширмовська Н.Г.

м. Івано-Франківськ
2018р.

Код:

```
struct node
{ infotype data;
  struct node *next;
};

typedef struct node *list;

void Init(list *head_list_ptr);
int Empty(list head_list);
void AddStart(list *head_list_ptr, infotype new_data);
void AddEnd(list *head_list_ptr, infotype new_data);
void PutAfter(list *node_prt, infotype new_data);
void PutBefore(list *node_prt, infotype new_data);
void Change(list *node1_ptr, list *node2_ptr);
list Find(list head_list, infotype search_data);
list FindBefore(list head_list, list node_ptr);
list FindAfter(list node_ptr);
list FindLast(list head_list);
void Delete(list *head_list_ptr, list *node_ptr);
void PrintStraight(list head_list);
void PrintReverse(list head_list);


#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
//#include "plist.h"


void Init(list *head_list_ptr)
{
      *head_list_ptr = NULL;
      return;
}

int Empty(list head_list)
{
      return head_list == NULL;
}


void AddStart(list *head_list_ptr, infotype new_data)
{
      list new_node;

      new_node = malloc(sizeof(struct node));
```

```
      new_node->data = new_data;
      new_node->next = *head_list_ptr;
      *head_list_ptr = new_node;
      return ;
}


void AddEnd(list *head_list_ptr, infotype new_data)
{
      list new_node;

      new_node = malloc(sizeof(struct node));
      new_node->data = new_data;
      new_node->next = NULL;
      if (Empty(*head_list_ptr)) *head_list_ptr = new_node;
      else
            FindLast(*head_list_ptr)->next = new_node;
      return ;
}


void PutAfter(list *node_prt, infotype new_data)
{
      list new_node = NULL;

      new_node = malloc(sizeof(struct node));
      new_node->data = new_data;
      new_node->next = (*node_prt)->next;
      (*node_prt)->next = new_node;
      return ;
}


void PutBefore(list *node_prt, infotype new_data)
{
      PutAfter(node_prt,new_data);
      Change(node_prt,&((*node_prt)->next));
      return ;
}


void Change(list *node1_ptr, list *node2_ptr)
{
      infotype tmp = (*node1_ptr)->data;
      (*node1_ptr)->data = (*node2_ptr)->data;
      (*node2_ptr)->data = tmp;
}
```

```
list Find(list head_list, infotype search_data)
{
      while ((head_list) && (head_list->data != search_data)) head_list =
head_list->next;
      return head_list;
}


list FindBefore(list head_list, list node)
{
      while ((head_list->next != node) && head_list) head_list =
head_list->next;
      return head_list;
}


list FindAfter(list node)
{
      return node->next;
}


list FindLast(list head_list)
{
      if (head_list) while (head_list->next) head_list = head_list->next;
      return head_list;
}


void Delete(list *head_list_ptr, list *node_ptr)
{
      list tmp , save_ptr = *node_ptr;

      assert(*head_list_ptr);
      assert(*node_ptr);

      if (*node_ptr == *head_list_ptr)
            *head_list_ptr = (*head_list_ptr)->next;
      else
            if (!((*node_ptr)->next))
            {

tmp = FindBefore(*head_list_ptr,*node_ptr);
                  tmp->next = NULL;
            }
            else
            {
```

```
                tmp = (*node_ptr)->next;
                (*node_ptr)->data = tmp->data;
                (*node_ptr)->next = tmp->next;
                save_ptr = tmp;
           };
      free(save_ptr);
      return ;
}


void PrintStraight(list head_list)
{
      while (head_list)
      {
           printf(printfspec,head_list->data);
           head_list = head_list->next;
      }
      printf("\n");
      return;
}


void PrintReverse(list head_list)
{
      if (head_list)
      {
           PrintReverse(head_list->next);
           printf(printfspec,head_list->data);
      }
      return;
}

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
//#include "plist.h"

list MergeLists(list list1, list list2);

void main()
{
      list l2 , fl;
      infotype x = 0;

      struct node a={3,NULL}, b={2,&a}, c={1,&b}, *l1 = &c;

      Init(&l2);
      printf("\nEnter elements in the second list:\n");
```

```
      do    {
            printf("Enter the new element(zero - break inputing elements):
");
            scanf("%d",&x);
            if (x)
              AddEnd(&l2,x);
      }
      while (x);

      PrintStraight(l1);
      PrintStraight(l2);

      x=l2->next->next->data;
      printf("%d  " , x);


      PutBefore(&(l1->next),4);
PrintStraight(l1);
      AddStart(&l1,6);
PrintStraight(l1);
      AddEnd(&l1,5);
      PrintStraight(l1);

      Delete(&l1,&l1);
      PrintStraight(l1);
      scanf("%d",&x);
      fl = Find(l1,x);
      Delete(&l1,&fl);
      PrintStraight(l1);

      Change(&l1,&(l1->next));
      PrintStraight(l1);

      printf("\nMerged lists:\n");
      PrintStraight(MergeLists(l1,l2));

      printf("\nPrinted merged lists in back order:\n");
      PrintReverse(MergeLists(l1,l2));
      getch();
      return;
}

list MergeLists(list list1, list list2)
{
      list mlist = NULL;

      while (list1)
      {
```

```
        AddEnd(&mlist,list1->data);
        list1 = list1->next;
    };

    while (list2 )
    {
        AddEnd(&mlist,list2->data);
        list2 = list2->next;
    };

    return mlist;
}
```

Висновок:

На даній лабораторній роботі я навчився працювати зі списками а також структурами даних.