Лабораторна робота № 10.

Тема: "Інформаційні динамічні структури"

Мета: Знайомство з динамічними інформаційними структурами на прикладі одно- і двонаправлених списків.

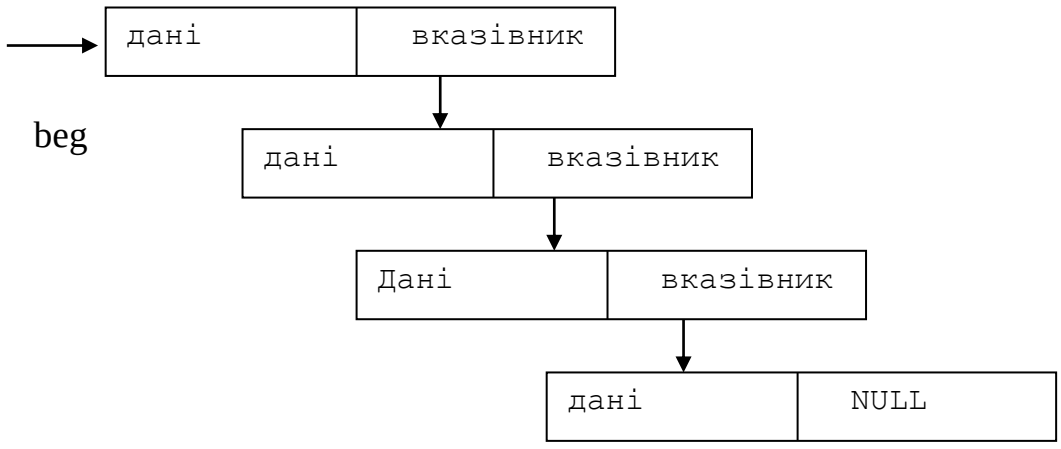
1. Короткі теоретичні відомості

У багатьох завданнях потрібно використовувати дані в яких конфігурація, розміри, склад можуть змінюватися в процесі виконання програми. Для їхнього представлення використовують динамічні інформаційні структури.

Найбільш проста інформаційна структура - це однозв'язний список, елементами якого є об’єкти структурного типу. Наприклад

```
struct ім'я_структурного_типу
{
елементи_структури;
struct ім'я_структурного_типу *вказівник;
}
```

У кожну структуру такого типу входить вказівник на об'єкт того ж типу, як і структура, що визначається.



Приклади:

1. Опис структури

```
struct point
{int key;
point* next;
};
```

Поле key містить інформаційну частину структури point, а поле next містить адресу наступного елемента списку.

2. Функція для формування однонапрвленого списку

```
point* make_point( int n)
{
point *first, *p;
first=NULL;
for (int i=n;i>0;i--)
{
p=new(point);
p->key=i;
p->next=first;
first=p;
}
return first;
}
```

Як параметр у функцію передається кількість елементів у списку, а результатом є вказівник на перший елемент цього списку. Вказівник p вказує на знову створюваний елемент. Для звертання до полів використовується операція доступу до елемента структури, з якою пов'язаний вказівник -> . Існує інша можливість звертання до поля динамічної структури: (*p).key або (*p).next. В інформаційне поле key заноситься порядковий номер елемента в списку. Додавання нових елементів здійснюється на початок списку.

3. Функція для друку однонаправленого списку

```
point* print_point(point*first)
```

```
{
if (first==NULL)return NULL;
point*p=first;
while(p!=NULL)
{
printf(“%d  ”, p->key);
p=p->next;
}
return first;
}
```

При друці сформованого списку здійснюється прохід за списком за допомогою допоміжної змінної p доти, поки вона не стане дорівнювати NULL.

2. Постановка завдання

Написати програму, у якій створюються динамічні структури й виконати їхню обробку у відповідності зі своїм варіантом.

Для кожного варіанту розробити такі функції:

1. Створення списку.
2. Додавання елемента в список (у відповідності зі своїм варіантом).
3. Знищення елемента зі списку (у відповідності зі своїм варіантом).
4. Друк списку.
5. Запис списку у файл.
6. Знищення списку.
7. Відновлення списку з файлу.

Порядок виконання роботи

1. Написати функцію для створення списку. Функція може створювати порожній список, а потім додавати в нього елементи.
2. Написати функцію для друку списку. Функція повинна передбачати вивід повідомлення, якщо список порожній.
3. Написати функції для знищення й додавання елементів списку у відповідності зі своїм варіантом.

4. Виконати зміни в списку й друк списку після кожної зміни.
5. Написати функцію для запису списку у файл.
6. Написати функцію для знищення списку.
7. Записати список у файл, знищити його й виконати друк (при друці повинне бути видане повідомлення "Список порожній").
8. Написати функцію для відновлення списку з файлу.
9. Відновити список і роздрукувати його.
10. Знищити список.

4. Варіанти завдань

1. Записи в лінійному списку містять ключове поле типу int. Сформувати однонаправлений список. Знищити з нього елемент із заданим номером, додати елемент із заданим номером;
2. Записи в лінійному списку містять ключове поле типу int. Сформувати однонаправлений список. Знищити з нього елемент із заданим ключем, додати елемент перед елементом із заданим ключем;
3. Записи в лінійному списку містять ключове поле типу int. Сформувати однонаправлений список. Знищити з нього К елементів, починаючи із заданого номера, додати елемент перед елементом із заданим ключем;
4. Записи в лінійному списку містять ключове поле типу int. Сформувати однонаправлений список. Знищити з нього елемент із заданим номером, додати К елементів, починаючи із заданого номера;
5. Записи в лінійному списку містять ключове поле типу int. Сформувати однонаправлений список. Знищити з нього К елементів, починаючи із заданого номера, додати К елементів, починаючи із заданого номера;
6. Записи в лінійному списку містять ключове поле типу int. Сформувати двонаправлений список. Знищити з нього елемент із заданим номером, додати елемент у початок списку.
7. Сформувати двонаправлений список. Знищити з нього перший елемент, додати елемент у кінець списку.

8. Записи в лінійному списку містять ключове поле типу int. Сформувати двонаправлений список. Знищити з нього елемент після елемента із заданим номером, додати К елементів у початок списку.
9. Записи в лінійному списку містять ключове поле типу int. Сформувати двонаправлений список. Знищити з нього К елементів перед елементом із заданим номером, додати К елементів у кінець списку.
10. Записи в лінійному списку містять ключове поле типу int. Сформувати двонаправлений список. Додати в нього елемент із заданим номером, знищити К елементів з кінця списку.
11. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений список. Знищити з нього елемент із заданим ключем, додати елемент із зазначеним номером.
12. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити з нього елементи, з однаковими ключовими полями. Додати елемент після елемента із заданим ключовим полем.
13. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити з нього К перших елементів. Додати елемент після елемента, що починається із зазначеного символу.
14. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити з нього К елементів із зазначеними номерами. Додати К елементів із зазначеними номерами.
15. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити К елементів з кінця списку. Додати елемент після елемента із заданим ключем.
16. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим ключем. Додати К елементів у кінець списку.

17. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим номером. Додати К елементів у початок списку.
18. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим ключем. Додати К елементів у початок списку.
19. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити К елементів із заданими номерами. Додати К елементів у початок списку.
20. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим ключем. Додати по К елементів на початок й в кінець списку.
21. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елементи перед і після елемента із заданим ключем. Додати по К елементів у початок й у кінець списку.
22. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим ключем. Додати К елементів перед елементом із заданим ключем.
23. Запису в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим ключем. Додати К елементів після елемента із заданим ключем.
24. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим номером. Додати по К елементів перед і після елемента із заданим ключем.
25. Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим ключем. Додати К елементів перед елементом із заданим номером.

4. Методичні вказівки

При програмній реалізації можна скористатись наступною прикладом.

Завдання.

Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елементи перед і після елемента із заданим ключем. Додати по К елементів у початок й у кінець списку.

```
#include<stdio.h>
#include<malloc.h>
//структура-вузол
typedef struct _Node {
    char* pc;
    struct _Node *next;
    struct _Node *prev;
} Node;
//зберігає розмір і вказівники на перший і останній елементи
typedef struct _DblLinkedList {
    size_t size;
    Node *head;
    Node *tail;
} DblLinkedList;
//створює екземпляр структури DblLinkedList...створює список
DblLinkedList* createDblLinkedList() {
    DblLinkedList *tmp = (DblLinkedList*)malloc(sizeof(DblLinkedList));
    tmp->size = 0;
    tmp->head = tmp->tail = NULL;
    return tmp;
}
//видаляє список
void deleteDblLinkedList(DblLinkedList **list) {
    Node *tmp = (*list)->head;
    Node *next = NULL;
    while (tmp) {
```

```
        next = tmp->next;
        free(tmp);
        tmp = next;
    }
    free(*list);
    (*list) = NULL;
}
//додаємо елемент в початок списку
void pushFront(DblLinkedList *list, char *data) {
    Node *tmp = (Node*) malloc(sizeof(Node));
    if (tmp == NULL) {
        exit(1);
    }
    tmp->pc = data;
    tmp->next = list->head;
    tmp->prev = NULL;
    if (list->head) {
        list->head->prev = tmp;
    }
    list->head = tmp;
    if (list->tail == NULL) {
        list->tail = tmp;
    }
    list->size++;
}
//видаляємо елемент з початку списку
void* popFront(DblLinkedList *list) {
    Node *prev;
    void *tmp;
    if (list->head == NULL) {
        exit(2);
```

```
    }
    prev = list->head;
    list->head = list->head->next;
    if (list->head) {
        list->head->prev = NULL;
    }
    if (prev == list->tail) {
        list->tail = NULL;
    }
    tmp = prev->pc;
    free(prev);
    list->size--;
    return tmp;
}
//вставка в кінець
void pushBack(DblLinkedList *list, char *value) {
    Node *tmp = (Node*) malloc(sizeof(Node));
    if (tmp == NULL) {
        exit(3);
    }
    tmp->pc = value;
    tmp->next = NULL;
    tmp->prev = list->tail;
    if (list->tail) {
        list->tail->next = tmp;
    }
    list->tail = tmp;
    if (list->head == NULL) {
        list->head = tmp;
    }
    list->size++;
```

```
}
 //видалення з кінця
void* popBack(DblLinkedList *list) {
    Node *next;
    void *tmp;
    if (list->tail == NULL) {
        exit(4);
    }
    next = list->tail;
    list->tail = list->tail->prev;
    if (list->tail) {
        list->tail->next = NULL;
    }
    if (next == list->head) {
        list->head = NULL;
    }
    tmp = next->pc;
    free(next);
    list->size--;
    return tmp;
}
//отримання n-го елемента
Node* getNth(DblLinkedList *list, size_t index) {
    Node *tmp = list->head;
    size_t i = 0;
    while (tmp && i < index) {
        tmp = tmp->next;
        i++;
    }
    return tmp;
}
```

```
//вставка чергового елемента
void insert(DblLinkedList *list, size_t index, char *value) {
    Node *elm = NULL;
    Node *ins = NULL;
    elm = getNth(list, index);
    if (elm == NULL) {
        exit(5);
    }
    ins = (Node*) malloc(sizeof(Node));
    ins->pc = value;
    ins->prev = elm;
    ins->next = elm->next;
    if (elm->next) {
        elm->next->prev = ins;
    }
    elm->next = ins;
    if (!elm->prev) {
        list->head = elm;
    }
    if (!elm->next) {
        list->tail = elm;
    }
    list->size++;
}
//видаляє вказаний елемент
void* deleteNth(DblLinkedList *list, size_t index) {
    Node *elm = NULL;
    void *tmp = NULL;
    elm = getNth(list, index);
    if (elm == NULL) {
        exit(5);
```

```
    }
    if (elm->prev) {
        elm->prev->next = elm->next;
    }
    if (elm->next) {
        elm->next->prev = elm->prev;
    }
    tmp = elm->pc;
    if (!elm->prev) {
        list->head = elm->next;
    }
    if (!elm->next) {
        list->tail = elm->prev;
    }
    free(elm);
    list->size--;
    return tmp;
}
void listprint(DblLinkedList *list)
{
  Node *p;
  p = list->head;
  if(p==NULL)
    printf("List empty!!!");
  else
  do {
    printf("%s ", p->pc);
    p = p->next;
  } while (p != NULL);
}
```

```
int main()
{
DblLinkedList *list = createDblLinkedList();
DblLinkedList *list1 = createDblLinkedList();
     listprint(list);
     printf("\n");
     FILE * fo;
       *fo;
      char* ptr;
      char *s;
     char  line[100];
     fo = fopen("test.txt", "rt");
     if( (fo=fopen("test.txt", "rt")) == 0 ) {
     // помилка!
     printf("Error open file!!!\n");
      }
    int i; char *x;
      while (!feof(fo))
        {
           fscanf(fo, "%s", line);
           x=line;
           pushBack(list, x);
          }
           printf("\n");
           listprint(list);
           printf("\n");
       fclose (fo);
    char *a, *b, *c, *d, *e, *f, *g, *h;
    a = "10";
    b = "20";
    c = "30";
```

```
 d = "40";
 e = "50";
 f = "60";
 g = "70";
 h = "80";
 pushFront(list, a);
 pushFront(list, b);
 pushFront(list, c);
 pushBack(list, d);
 pushBack(list, e);
 pushBack(list, f);
 listprint(list);
 printf("\n");
 printf("length %d\n", list->size);
 printf("nth 3 %s\n", ((char*)(getNth(list, 2))->pc));
 printf("nth 6 %s\n", ((char*)(getNth(list, 5))->pc));
 printf("popFront %s\n", ((char*)popFront(list)));
 listprint(list);
 printf("\n");
 int ind;
printf("Input index of element: ");
scanf("%d", &ind);
if((ind>1)&&(ind<(list->size)))
   {
 deleteNth(list, ind-2);
 deleteNth(list, ind-1);
 listprint(list);
 printf("\n");
 }
 else
 {
```

```
            printf("No Delete!!!\n");
    }
    int k;
    printf("Input count of element: ");
    scanf("%d", &k);
    for(int i=0; i<k; i++)
    {
     pushBack(list, g);
     pushFront(list, h);
    }
     listprint(list);
     printf("\n");
      printf("popFront %s\n", ((char*)popFront(list)));
      listprint(list);
      printf("\n");
      printf("head %s\n", ((char*)(list->head->pc)));
      printf("tail %s\n", ((char*)(list->tail->pc)));
    printf("popBack %s\n", ((char*)popBack(list)));
    printf("popBack %s\n", ((char*)popBack(list)));
    printf("length %d\n", list->size);
    listprint(list);
    printf("\n");
    insert(list, 1, g);
    listprint(list);
    printf("\n");
    deleteNth(list, 0);
    listprint(list);
    printf("\n");
    deleteDblLinkedList(&list);
    return 0;
}
```

5.Зміст звіту

1. Постановка завдання.
2. Функції для роботи зі списком.
3. Функція main().

Результати виконання р