Лабораторна робота №19.
Динамічні списки.

- **Вимоги**
- **Розробник**
  - Гладков Костянтинт Сергіович
  - Студент групи КІТ-320;
  - 23-jan-2021.
- **Основне завдання**

Завдання на оцінку "задовільно":

На базі попередньо розробленого функціоналу по роботі з прикладною областю сформувати односпрямований список елементів розробленої структури. Реалізувати наступні функції роботи зі списком:

- читання даних з файлу, використовуючи функцію fscanf;
- запис даних у файл, використовуючи функцію fprintf;
- вивід вмісту списку на екран;
- пошук у списку об'єкта за заданим критерієм;
- додавання об'єкта у кінець списку;
- видалення об'єкта зі списку.

Завдання на оцінку "добре":

- виконати завдання на "задовільно";
- реалізувати діалоговий режим спілкування з користувачем за допомогою меню;
- сортування вмісту списку за одним з критеріїв. При цьому дозволяється міняти місцями не вказівники на об'єкти, а вміст об'єктів (за допомогою третьої змінної);
- переробити метод додавання з можливістю вставлення додаткового елементу після будь-якого елементу списку;
- продемонструвати відсутність витоків пам'яті;
- розробити модульні тести, що демонструють коректність роботи реалізованих функцій
- проект має складатися мінімум з 6 файлів (main.c, test.c, list.h, list.c, data.c, data.h)

Завдання на оцінку "відмінно":

- виконати завдання на "добре", але замість односпрямованого списку треба використовувати доспрямоваинй;
- Реалізувати сортування вмісту списку за одним з критеріїв. При цьому обов'язково забезпечити, щоб обмін місцями об'єктів здійснювався шляхом обміну їх покажчиків.

- **Опис роботи**

Програма працює шляхом введення даних від користувача з файлу і їх подальшого аналізу з виведенням на екран та в файл результату. Також в кінці відбувається тест.

- **Функціональне призначення**

Дана програма працює за допомогою використання динамічної пам'яті, а точніше е е маніпуляцією, також використовується файли.

- **Опис логічної структури**

Дана програма може бути використана для створення динамічного списку, двонаправленого, який буде зберігати собі дані і буде легкий для підправлені (видалення і введення даних)

Функція main містить в собі виклик функцій, а також корисних властивостей для функцій

Функція append збільшує розмір списку

Функція push вставляє в кінець елемент

Функція insertAfter вставляє елемент після певного індексу

Функція printList показує наш список

Функція bubbleSort сортровка бульбашкою

Функція swap служить для сортування

Функція test_append перевіряє функцію

Функція print_node виводить конкретний елемент

Функція insertAtTheBegin вставляє елемент в початок

Функція printList_reverse показує наш список в зворотному порядку

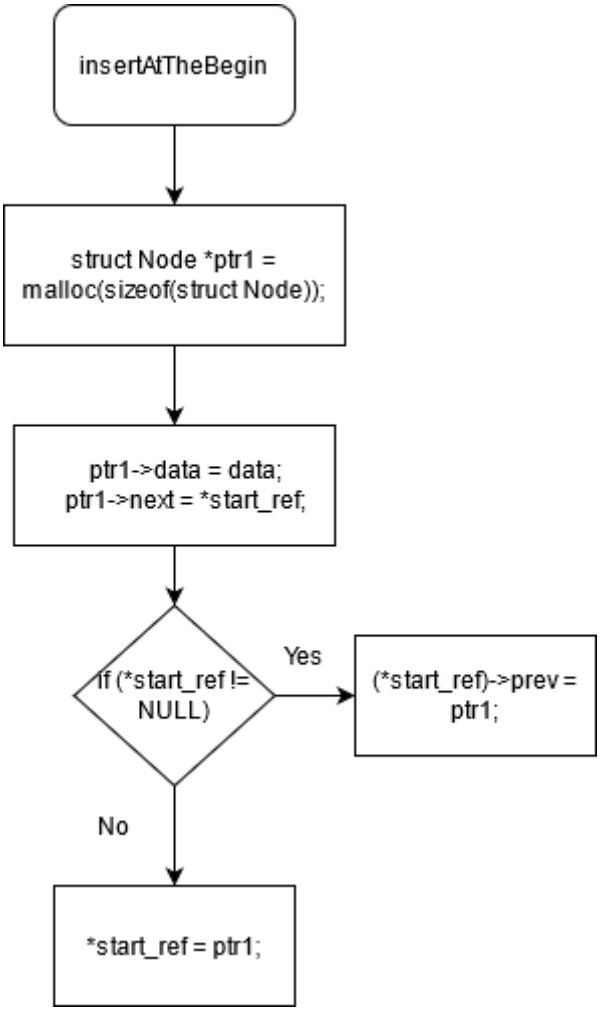

рис. 1 - insertAtTheBegin

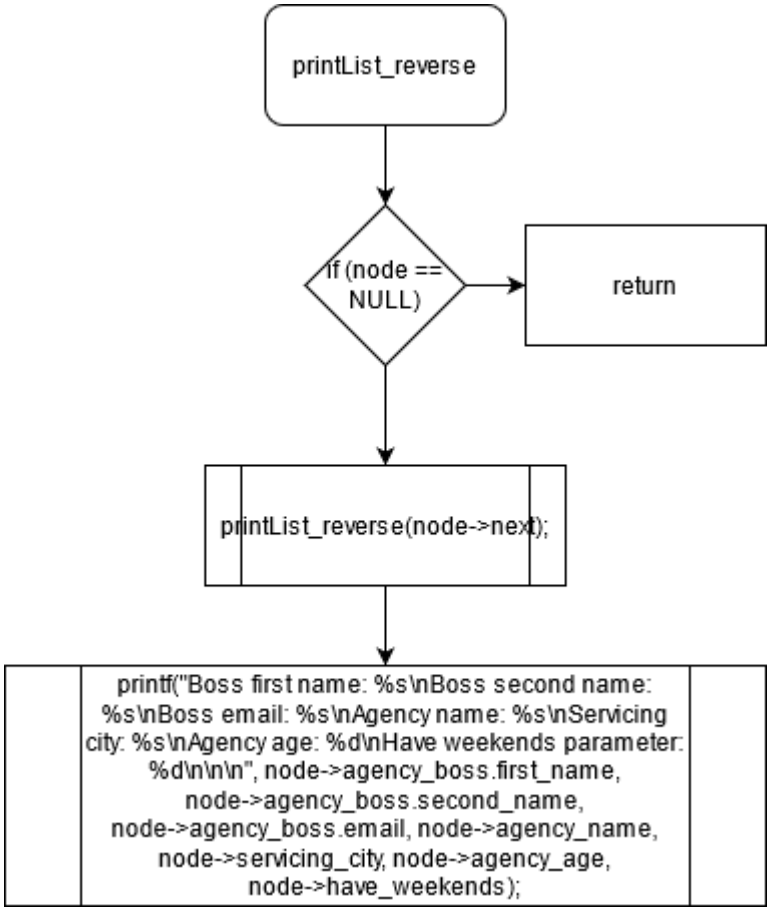

рис. 2 - printList_reverse

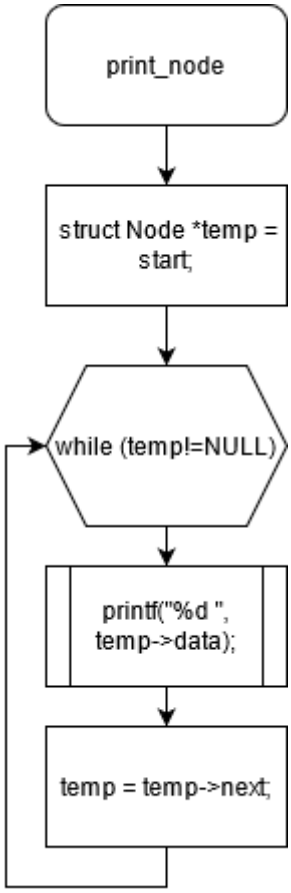

рис. 3 - print_node

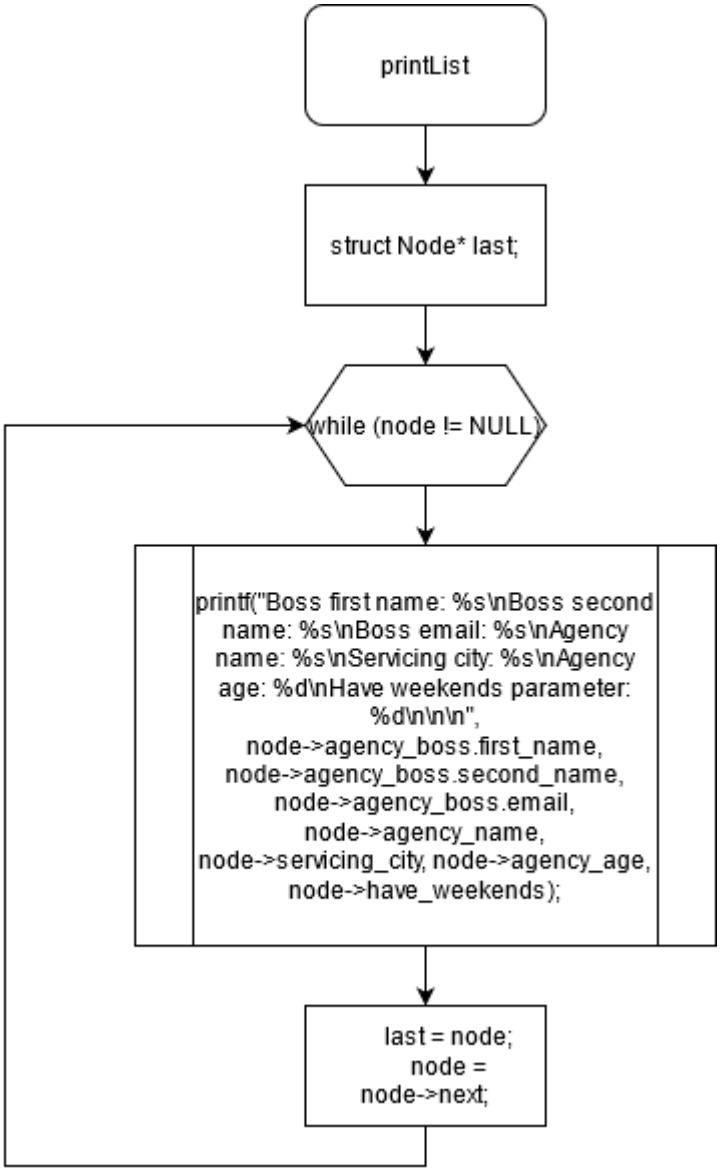

рис. 4 - printList

- **Важливі елементи програми**

```
struct Agency* new_node = malloc(sizeof(struct Agency));

struct Agency* last = *head_ref;

strcpy(new_node->agency_boss.first_name, "x");
strcpy(new_node->agency_boss.second_name, "x");
strcpy(new_node->agency_boss.email, "x");
```

```
strcpy(new_node->agency_name, "x");
strcpy(new_node->servicing_city, "x");
new_node->agency_age = 5;
new_node->have_weekends = 0;

new_node->next = NULL;

if (*head_ref == NULL) {
    new_node->prev = NULL;
    *head_ref = new_node;
    return;
}

while (last->next != NULL)
    last = last->next;

last->next = new_node;

new_node->prev = last;
```

```
struct Node *temp = start;
printf("\n");
while (temp!=NULL)
{
    printf("%d ", temp->data);
    temp = temp->next;
}
```

```
struct Node *ptr1 = malloc(sizeof(struct Node));
ptr1->data = data;
ptr1->next = *start_ref;
if (*start_ref != NULL){
    (*start_ref)->prev = ptr1;
}
*start_ref = ptr1;
```

- **Варіанти використання**

Дана програма може бути використана для роботи з динамічними списками.

Висновки

У даній лабораторній роботі був пріобрітен досвід роботи з динамічними списками.