Фаховий коледж ракетно-космічного машинобудування

Дніпровського національного університету імені Олеся Гончара

ЗВІТ

з лабораторних робіт

з дисципліни «Технології (Програмування)»

Спеціальність 123 Комп’ютерна інженерія

Група КС-21-1

Виконав А.С. Василенко

Перевірив М.М. Гапоненко

2022-2023

ЗМІСТ

Лабораторна робота № 18

ЛАБОРАТОРНА РОБОТА № 18

Мета: Придбати практичні навички роботи з Heap – пам’яттю при складанні та налагодженні програм створення та обробки однозв’язних списків даних.

Хід роботи

1 Постановка задачі.

Загальна постановка завдання:

Скласти та налагодити програму обробки однозв’язного списку за алгоритмом згідно Вашого варіанту. Програма повинна задовольняти наступним вимогам:

− організувати користувацьке меню, яке повинно містити наступні пункти:

1. Формування списку.

2. Перегляд вмісту списку.

3. Обробка списку.

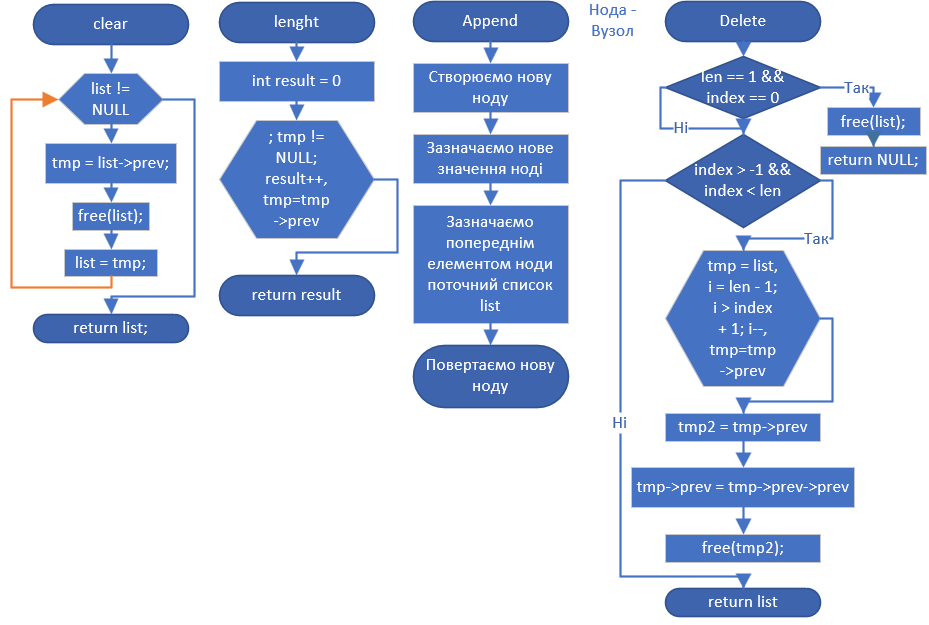
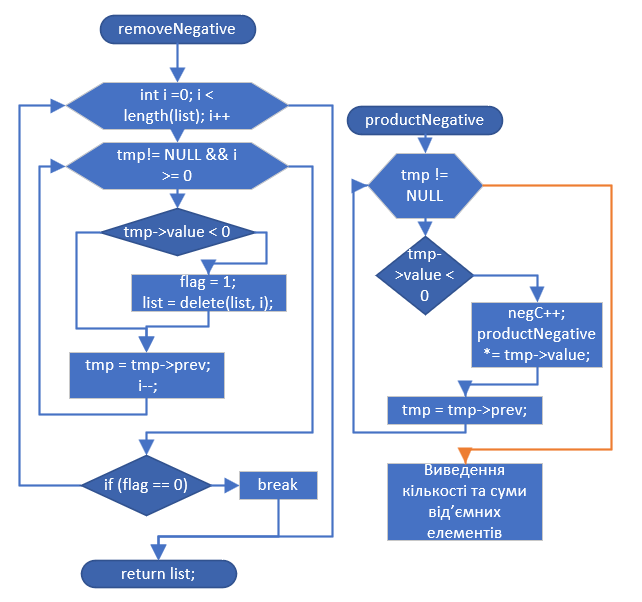
4. Видалення списку.

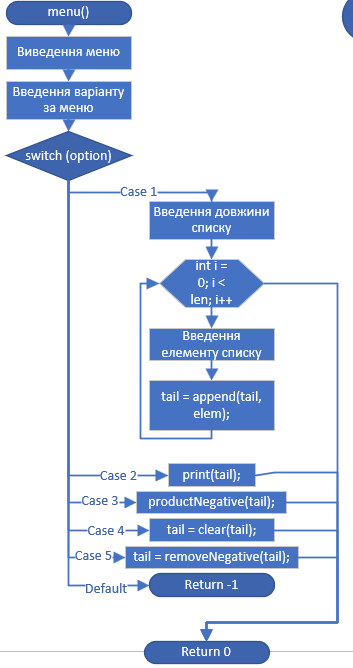
− забезпечити коректне введення користувачем вхідних даних;

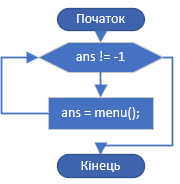
− при обробці списку враховувати, що шукані елементи можуть бути відсутні. В цьому випадку вивести користувачеві відповідне повідомлення;

− введення та виведення вхідних та вихідних даних повинно містити необхідні для користувача повідомлення.

Створити однозв’язний список дійсних чисел введенням нового елементу в початок списку. Список повинен вміщати як додатні, так і від’ємні числа. Знайти та вивести добуток та підрахувати кількість від’ємних елементів списку. Видалити із списку всі від’ємні елементи.

2 Блок-схема алгоритму рішення задачі. 





3 Текст програми мовою програмування С++.

#include <stdio.h>

#include <malloc.h>

#include <windows.h>

// structure to perform list node

struct \_Node {

    int value;           // element value

    struct \_Node\* prev;  // previous element pointer

} \*tail=NULL; // global list variable

// calculate length of list

int length(struct \_Node\* *list*)

{

    int result = 0;

    if (*list* != NULL) {

        struct \_Node\* tmp = *list*;

        for (; tmp != NULL; result++, tmp=tmp->prev);

    }

    return result;

}

// adds new element to the end of the list

struct \_Node\* append(struct \_Node\* *list*, int *value*)

{

    struct \_Node\* newNode = (struct \_Node\*)malloc(sizeof(struct \_Node));

    newNode->value = *value*;

    newNode->prev = *list*;

    return newNode;

}

// delete element by index

struct \_Node\* delete(struct \_Node\* *list*, int *index*)

{

    int len = length(*list*);

    if (len == 1 && *index* == 0) {

        free(*list*);

        return NULL;

    }

    if (*index* > -1 && *index* < len) {

        struct \_Node \*tmp, \*tmp2;

        int i;

        for (tmp = *list*, i = len - 1; i > *index* + 1; i--, tmp=tmp->prev);

        tmp2 = tmp->prev;

        tmp->prev = tmp->prev->prev;

        free(tmp2);

    }

    return *list*;

}

// remove all elements from the list

struct \_Node\* clear(struct \_Node\* *list*)

{

    printf("\n      | Список Очищенно |\n");

    struct \_Node\* tmp = *list*;

    while (*list* != NULL) {

        tmp = *list*->prev;

        free(*list*);

*list* = tmp;

    }

    return *list*;

}

// forward index-based element printing

void print(struct \_Node\* *list*)

{

    if (*list* != NULL) {

        print(*list*->prev);

        printf("%d ", *list*->value);

    }

}

void productNegative(struct \_Node\* *list*) {

    int negC = 0; // negative number counter

    int productNegative = 1;

    struct \_Node\* tmp = *list*;

    while (tmp != NULL) {

        if (tmp->value < 0) {

            negC++;

            productNegative \*= tmp->value;

        }

        tmp = tmp->prev;

    }

    printf("      | Negative numbers counter: %d \n       |proudct of negative numbers: %d", negC, productNegative\*(-1));

}

struct \_Node\* removeNegative(struct \_Node\* *list*) {

    struct \_Node\* tmp;

    int i;

    int flag;

    for (int i =0; i < length(*list*); i++) {

    flag = 0;

    i=length(*list*)-1;

    tmp = *list*;

        while (tmp!= NULL && i >= 0) {

            if (tmp->value < 0) {

                flag = 1;

*list* = delete(*list*, i);

                printf("\b");

                print(*list*);

                i++;

            }

            tmp = tmp->prev;

            i--;

        }

        if (flag == 0) break;

    }

    return *list*;

}

int menu() {

    int option=-1;

    int len;

    printf("\n\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \n1.    Формування списку.\n2.  Перегляд вмісту списку.\n3. Добуток та кількість від'ємних елементів списку.\n4.    Видалення списку.\n5.   Видалити із списку всі від’ємні елементи.\n6.  Вийти з програми\n\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*\n");

    printf("Введіть варіант відповіді: ");

    scanf("%d", &option);

    switch (option)

    {

    case 1: // Формування списку

        printf("Введіть довжину списку: ");

        scanf("%d", &*len*);

        int elem;

        printf("Введіть елементи списку\n");

        for (int i = 0; i < len; i++) {

            printf("      [%d]: ", i);

            scanf("%d", &elem);

            tail = append(tail, elem);

        }

        break;

    case 2: // Перегляд вмісту списку

        printf("      - Список -\n------> ");

        print(tail);

        break;

    case 3: // Добуток та кількість від'ємних елементів списку.

        productNegative(tail);

        break;

    case 4: // Видалення списку.

        tail = clear(tail);

        break;

    case 5: //Видалити із списку всі від’ємні елементи.

        tail = removeNegative(tail);

        break;

    default:

        return -1;

    }

    return 0;

}

int main()

{

    SetConsoleCP(65001);

    SetConsoleOutputCP(65001);

    int ans = 0;

    while (ans != -1) {

        ans = menu();

    }

    return 0;

}

4 Копія вікна виконання програми

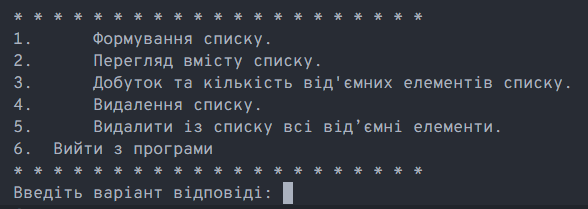


Рисунок 1 Меню

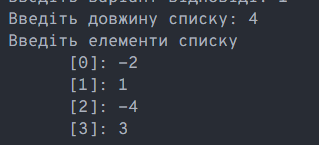


Рисунок 2 (1) Формування списку



Рисунок 3 (2) Перегляд списку



Рисунок 4 (3) Добуток та кількість від’ємних елементів списку



Рисунок 5 (5) Видалення всіх від’ємних елементів списку   
(Перегляд вмісту списку після видалення від’ємних елементів )

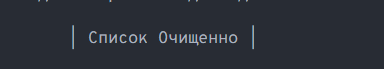




Рисунок 6 (4) Видалення списку



Рисунок 7 (6) Вихід з програми

5 Висновок

В результаті виконання лабораторної роботи опрацьовано теоретичний матеріал за темами:

Однозв’язні списки

Вказівники на структури