Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота № 2.2

з дисципліни
«Алгоритми і структури даних»

Виконав:

студент групи ІМ-22
Кушнір Микола Миколайович
номер у списку групи: 13

Перевірила:

Молчанова А. А.

Київ 2023

Постановка задачі

1. Створити список з ***n*** (***n*** > 0) елементів (***n*** вводиться з клавіатури), якщо інша кількість елементів не вказана у конкретному завданні.

2. Тип ключів (інформаційних полів) задано за варіантом.

3. Значення елементів списку взяти самостійно такими, щоб можна було продемонструвати коректність роботи алгоритму програми. Введення значень елементів списку можна виконати довільним способом (випадкові числа, формування значень за формулою, введення з файлу чи з клавіатури).

4. Вид списку (черга, стек, дек, прямий однозв'язний лінійний список, обернений однозв'язний лінійний список, двозв'язний лінійний список, однозв'язний кільцевий список, двозв'язний кільцевий список) вибрати самостійно з метою найбільш доцільного рішення поставленої за варіантом задачі.

5. Виконати над створеним списком дії, вказані за варіантом, та коректне звільнення пам'яті списку.

6. При виконанні заданих дій, виводі значень елементів та звільненні пам'яті списку вважати, що довжина списку (кількість елементів ***n*** чи ***2n***) невідома на момент виконання цих дій.

7. Повторювані частини алгоритму необхідно оформити у вигляді процедур або функцій (для створення, обробки, виведення та звільнення пам'яті списків) з передачею списку за допомогою параметра(ів).

Завдання для варіанту 13

Ключами елементів списку є цілі числа. Переставити елементи списку так, щоб спочатку розташовувались додатні, потім нульові, а за ними від'ємні елементи, не використовуючи додаткових структур даних, крім простих змінних (тобто «на тому ж місці»).

Посилання на репозиторій з кодом лабораторної роботи

Текст програми

Вміст файлу WorkingWithLinkedList.h

```
#include <stdlib.h>
#include "InputValidation.h"
#include "PrimitiveTableOutput.h"

typedef struct linkedList {
    int key;
    struct linkedList *next;
} linkList;

linkList *initList(int data) {
    linkList *firstP;
    firstP = malloc(sizeof(linkList));
    firstP->key = data;
    firstP->next = NULL;
    return firstP;
}

linkList *addItem(linkList *listP, int data) {
    linkList *newP;
    newP = malloc(sizeof(linkList));
    newP->key = data;
    newP->next = listP;
    return newP;
}

linkList *createManually(int numOfItems) {
    printf("Enter a value of key for item 1\n> ");
    validRes data = askInt();
    if (!data.isValid) return NULL;
    linkList *currentP = initList(data.value);
    int counter = 2;
    for (int i = 1; i < numOfItems; ++i) {
        printf("Enter a value of key for item %d\n> ", counter);
        data = askInt();
        while (!data.isValid) {
            printf("Invalid input! Try again!\n> ");
            data = askInt();
        }
        ++counter;
        currentP = addItem(currentP, data.value);
    }
    return currentP;
}

linkList *createRandomly(int numOfItems) {
    int data = (rand() % 201);
    if (data > 100) data -= 201; // -100 <= data <= 100
    linkList *currentP = initList(data);
    for (int i = 1; i < numOfItems ; ++i) {
        data = (rand() % 201);
        if (data > 100) data -= 201;
        currentP = addItem(currentP, data);
    }
    return currentP;
}
```

```
linkList *createList() {
    printf("How many items should be created?\n> ");
    validRes numOfItems = askInt();
    if (!(numOfItems.isValid && numOfItems.value > 0)) {
        return NULL;
    }
    printf("Do you want to manually enter the values of\n"
           "keys for items or set pseudo-random keys?\n"
           "* 1st variant - enter 1 and press 'ENTER' *\n"
           "* 2nd variant - enter 2 and press 'ENTER' *\n> ");
    validRes input = askInt();
    linkList *listP;
    switch (input.value) {
        case 1:
            listP = createManually(numOfItems.value);
            break;
        case 2:
            listP = createRandomly(numOfItems.value);
            break;
        default:
            listP = NULL;
    }
    system("cls");
    return listP;
}

linkList *sortList(linkList *listP) {
    linkList *previousP = listP;
    linkList *currentP = previousP->next;
    while (listP->key <= 0 && currentP) {
        if (currentP->key >= 0) {
            previousP->next = currentP->next;
            currentP->next = listP;
            listP = currentP;
        } else previousP = currentP;
        currentP = previousP->next;
    }
    linkList *middleP = listP;
    while (currentP) {
        if (currentP->key > 0) {
            previousP->next = currentP->next;
            currentP->next = listP;
            listP = currentP;
        } else if (currentP->key == 0 && previousP->key < 0) {
            previousP->next = currentP->next;
            currentP->next = middleP->next;
            middleP->next = currentP;
        } else previousP = currentP;
        currentP = previousP->next;
    }
    return listP;
}

void printList(linkList *listP) {
    int numOfCols = 4;
    int lenOfCols[] = {5, 18, 13, 18};
    linkList *currentP = listP;
    int counter = 1;
    printTableLine(numOfCols, lenOfCols, 1);
```

```
    printf("%c No. %c   item pointer   %c     key     %c   next pointer
%c\n",
           179, 179, 179, 179, 179);
    while (currentP) {
        printTableLine(numOfCols, lenOfCols, 2);
        printf("%c %2d  %c %p %c %11d %c %p %c\n",
               179, counter, 179, currentP, 179, currentP->key, 179,
currentP->next, 179);
        currentP = currentP->next;
        ++counter;
    }
    printTableLine(numOfCols, lenOfCols, 3);
}

void deleteList(linkList *listP) {
    int numOfCols = 3;
    int lenOfCols[] = {26, 13, 18};
    linkList *currentP = listP;
    printTableLine(numOfCols, lenOfCols, 1);
    printf("%c   current list pointer   %c     key     %c   next pointer
%c\n",
           179, 179, 179, 179);
    while (currentP) {
        printTableLine(numOfCols, lenOfCols, 2);
        printf("%c     %p     %c %11d %c %p %c\n",
               179, currentP, 179, currentP->key, 179, currentP->next,
179);
        linkList *temporaryP = currentP;
        currentP = currentP->next;
        free(temporaryP);
    }
    printTableLine(numOfCols, lenOfCols, 2);
    printf("%c     %p     %c             %c                  %c\n",
           179, currentP, 179, 179, 179);
    printTableLine(numOfCols, lenOfCols, 3);
    listP = NULL;
}
```

Вміст файлу InputValidation.h

```
#define NEW_LINE_ASCII 10
#define SPACE_ASCII 32
#define DASH_ASCII 45
const int digitsInASCII[] = {48, 49, 50, 51, 52, 53, 54, 55, 56, 57};

#define MAX_INT 2147483647
#define MIN_INT (-2147483648)

int checkDigit(int data) {
    int isDigit = 0, i;
    for (i = 0; i < 10; ++i) {
        if (data == digitsInASCII[i]) isDigit = 1;
    }
    return isDigit;
}

int convertASCIIToInt(int data) {
    return (int)(data - 48);
}
```

```
typedef struct validationResult {
    int value;
    int isValid;
} validRes;

validRes askInt() {
    validRes data;
    data.isValid = 1;
    data.value = 0;
    int isNegative = 0;
    long long accumulator = 0;
    int input = getchar();
    while (input == SPACE_ASCII) input = getchar();
    if (input == DASH_ASCII) isNegative = 1;
    else if (checkDigit(input)) accumulator += convertASCIIToInt(input);
    else data.isValid = 0;
    while (input != NEW_LINE_ASCII && data.isValid) {
        input = getchar();
        if (checkDigit(input)) {
            accumulator = (accumulator * 10) + convertASCIIToInt(input);
            if (accumulator < MIN_INT || accumulator > MAX_INT) {
                data.isValid = 0;
            }
        } else if (input != SPACE_ASCII && input != NEW_LINE_ASCII) {
            data.isValid = 0;
        }
    }
    if (data.isValid) {
        data.value = (int) accumulator;
        if (isNegative) data.value *= -1;
    }
    while (input != NEW_LINE_ASCII) input = getchar();
    return data;
}
```

Вміст файлу PrimitiveTableOutput.h

```
#define FIRST_LINE 1
int charsOfFirstLine[] = {218, 196, 194, 191};

#define MIDDLE_LINE 2
int charsOfMiddleLine[] = {195, 196, 197, 180};

#define LAST_LINE 3
int charsOfLastLine[] = {192, 196, 193, 217};

void printTableLine(int numOfCols, int lenOfCols[numOfCols], int
typeOfLine) {
    int *pointer;
    switch (typeOfLine) {
        case FIRST_LINE:
            pointer = charsOfFirstLine;
            break;
        case MIDDLE_LINE:
            pointer = charsOfMiddleLine;
            break;
        case LAST_LINE:
            pointer = charsOfLastLine;
```

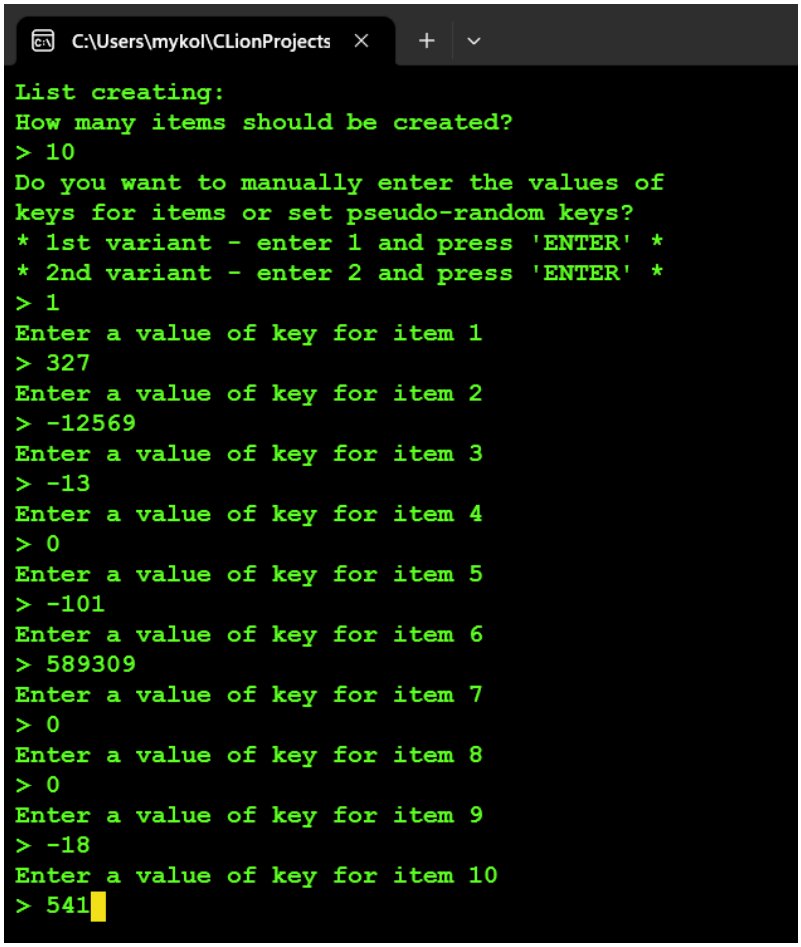
```
    }
    int symbols[4];
    int i, j;
    for (i = 0; i < 4; ++i) symbols[i] = pointer[i];

    printf("%c", symbols[0]);
    for (i = 0; i < numOfCols; ++i) {
        for (j = 0; j < lenOfCols[i]; ++j) printf("%c", symbols[1]);
        if ((i + 1) != numOfCols) printf("%c", symbols[2]);
    }
    printf("%c\n", symbols[3]);
}
```

Вміст файлу main.c

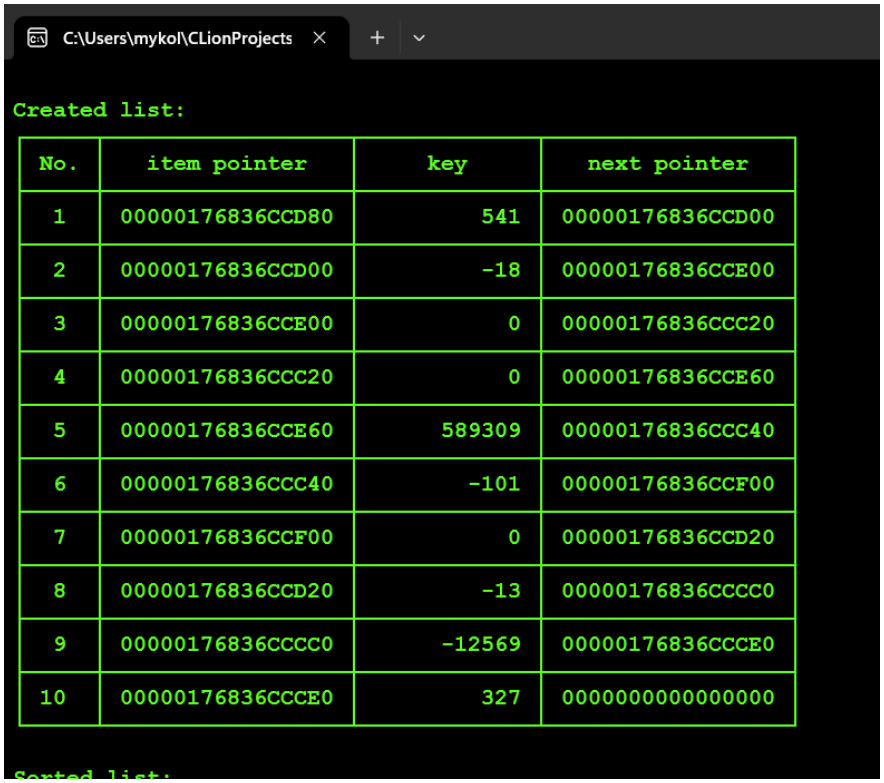
```
#include <stdio.h>
#include "WorkingWithLinkedList.h"

int main() {
    printf("List creating:\n");
    linkList *listPointer = createList();

    if (listPointer) {
        printf("\nCreated list:\n");
        printList(listPointer);

        printf("\nSorted list:\n");
        listPointer = sortList(listPointer);

        printList(listPointer);

        printf("\nMemory freeing:\n");
        deleteList(listPointer);
    } else {
        printf("Invalid input!");
    }

    printf("\n* press 'ENTER' to exit the program *\n");
    getchar();
    return 0;
}
```

Результати тестування програми

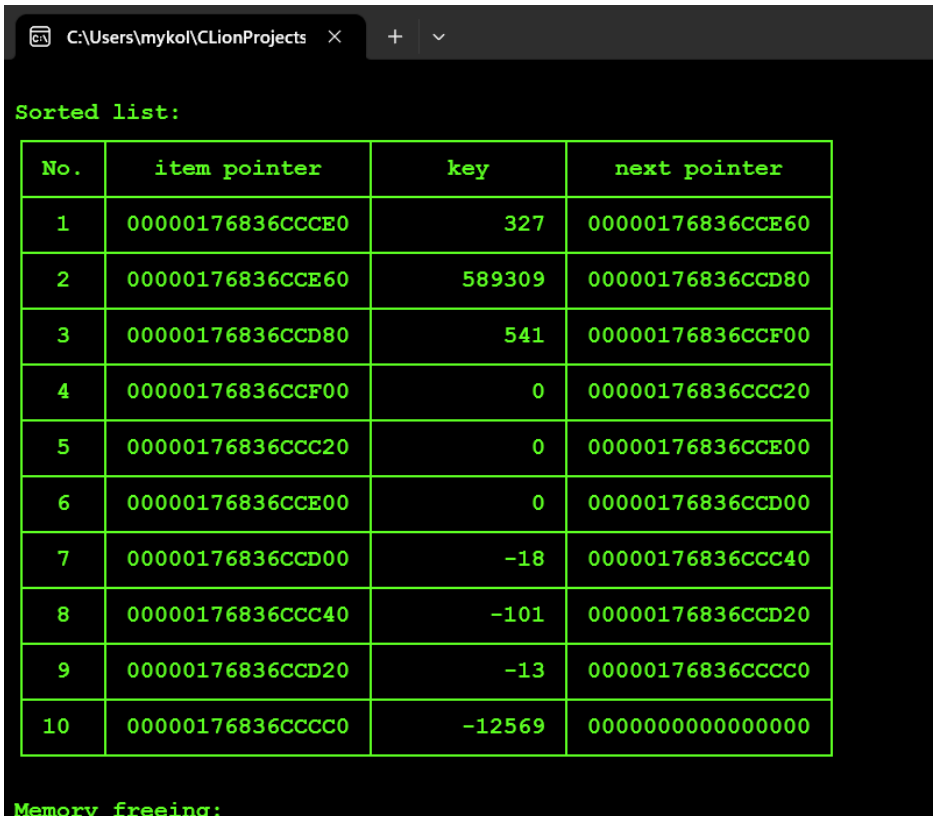
Вхідні дані



Створений список

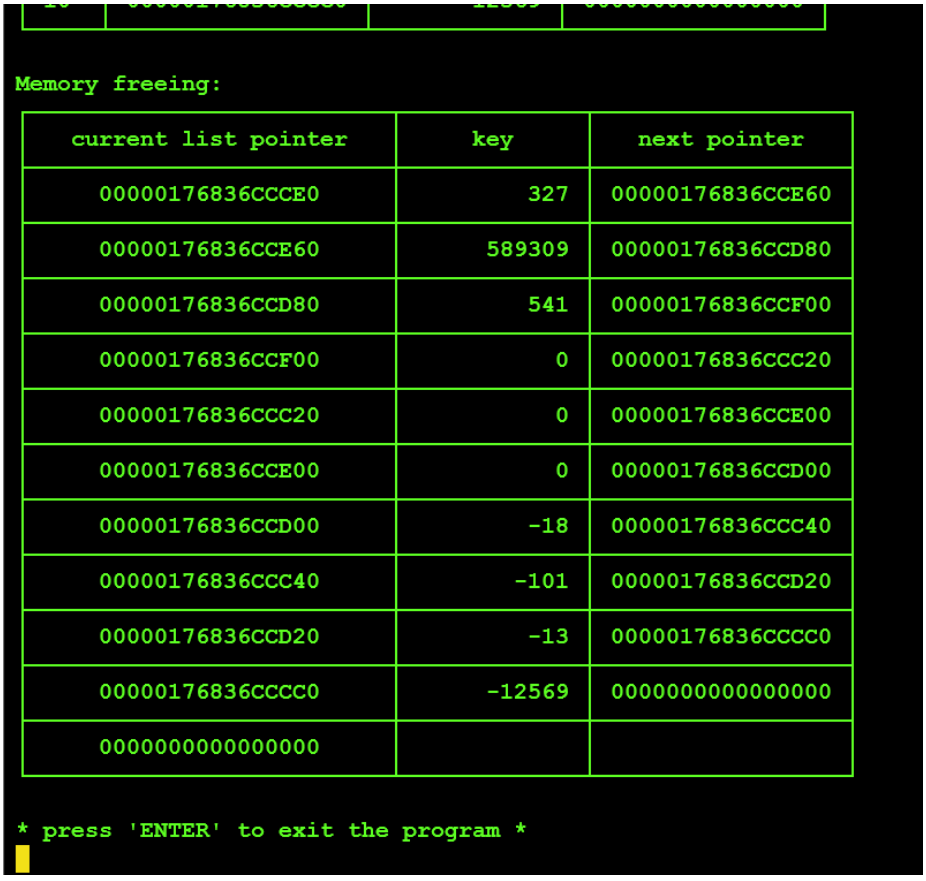


Відсортований список



No.	item pointer	key	next pointer
1	00000176836CCCE0	327	00000176836CCE60
2	00000176836CCE60	589309	00000176836CCD80
3	00000176836CCD80	541	00000176836CCF00
4	00000176836CCF00	0	00000176836CCC20
5	00000176836CCC20	0	00000176836CCE00
6	00000176836CCE00	0	00000176836CCD00
7	00000176836CCD00	-18	00000176836CCC40
8	00000176836CCC40	-101	00000176836CCD20
9	00000176836CCD20	-13	00000176836CCCC0
10	00000176836CCCC0	-12569	0000000000000000

Звільнення пам’яті



current list pointer	key	next pointer
00000176836CCCE0	327	00000176836CCE60
00000176836CCE60	589309	00000176836CCD80
00000176836CCD80	541	00000176836CCF00
00000176836CCF00	0	00000176836CCC20
00000176836CCC20	0	00000176836CCE00
00000176836CCE00	0	00000176836CCD00
00000176836CCD00	-18	00000176836CCC40
00000176836CCC40	-101	00000176836CCD20
00000176836CCD20	-13	00000176836CCCC0
00000176836CCCC0	-12569	0000000000000000
0000000000000000		