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 №8

з дисципліни
«Алгоритми і структури даних»

Виконав:
студент групи ІМ-42
Туров Андрій Володимирович
номер у списку групи: 28

Перевірив:
Сергієнко А. М.

Київ 2025

Завдання

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

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

3. 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 розв'язку поставленої за варіантом задачі.

4. Створити функції (або процедури) для роботи зі списком (для створення, обробки, додавання чи видалення елементів, виводу даних зі списку в консоль, звільнення пам'яті тощо).

5. 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).

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

7. *При виконанні заданих дій, виводі значень елементів та звільненні пам'яті списку вважати, що довжина списку (кількість елементів) невідома на момент виконання цих дій.* Тобто, не дозволяється зберігати довжину списку як константу, змінну чи додаткове поле.

Варіант 28

Ключами елементів списку є цілі ненульові числа. Кількість елементів списку n повинна бути кратною 10-ти, а елементи у початковому списку розташовуватись із чергуванням знаків. Перекомпонувати список, змінюючи порядок чисел всередині кожного десятка елементів так, щоб спочатку йшли від'ємні числа цього десятка елементів, а за ними — додатні, не використовуючи додаткових структур даних, крім простих змінних (тобто «на тому ж місці»).

Текст програм

```c
#include "input.h"
#include "linkedlist.h"
#include <stddef.h>
```

```c
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

LinkedList *get_input(const int32_t argc, char **argv) {
  LinkedList *head = LinkedList_new(0);
  LinkedList *tail = head;

  if (argc ⩽ 1) {
    printf("No input received. Generating random sequence...\n");
    int8_t sign = (rand() % 2) ? 1 : -1;
    for (int32_t i = (rand() % 5 + 1) * GROUP_SIZE; i > 0; i--) {
      int32_t element = rand() % 100;
      tail = LinkedList_push(tail, sign * element);
      sign *= -1;
    }
  } else {
    if ((argc - 1) % GROUP_SIZE ≠ 0) {
      printf("The amount of elements has to be a multiple of
       ↪  %d!", GROUP_SIZE);
      return NULL;
    }
    // TODO: sign validation
    for (int32_t i = 1; i < argc; i++) {
      tail = LinkedList_push(tail, atoi(argv[i]));
    }
  }
  return head->next;
}
```

Файл 1: input.c

```c
#pragma once

#include "linkedlist.h"
#include <stdint.h>

#define GROUP_SIZE 10

LinkedList *get_input(const int32_t argc, char **argv);
```

Файл 2: input.h

```c
#include "linkedlist.h"
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
```

```c
LinkedList *LinkedList_new(const int32_t value) {
  LinkedList *head = (LinkedList *)malloc(sizeof(LinkedList));
  head->value = value;
  head->next = (struct LinkedList *)NULL;
  return head;
}

LinkedList *LinkedList_push(LinkedList *list, const int32_t
↪  value) {
  LinkedList *next = LinkedList_new(value);
  list->next = (struct LinkedList *)next;
  return next;
}

void LinkedList_display(LinkedList *list) {
  LinkedList *node = list;
  size_t index = 0;
  while (node ≠ NULL) {
    printf("%d ", node->value);
    index++;
    // NOTE: should made the output clearer, but isn't suitable
    ↪  for LaTeX
    //
    // if (index % 20 == 0) {
    //   printf("\033[0m");
    // } else if (index % 10 == 0) {
    //   printf("\033[38;5;245m");
    // }
    node = node->next;
  }
  // printf("\033[0m");
  printf("\n");
}

void LinkedList_free(LinkedList *list) {
  if (list == NULL) {
    return;
  }
  LinkedList_free(list->next);
  free(list);
}
```

Файл 3: linkedlist.c

```c
#pragma once

#include <stdint.h>

typedef struct LinkedList {
```

```
    int32_t value;
    struct LinkedList *next;
} LinkedList;

LinkedList *LinkedList_new(const int32_t value);
LinkedList *LinkedList_push(LinkedList *list, const int32_t
 ↪  value);
void LinkedList_display(LinkedList *list);
void LinkedList_free(LinkedList *list);
```

Файл 4: linkedlist.h

```
#include "input.h"
#include "linkedlist.h"
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

typedef struct {
  LinkedList *head;
  LinkedList *tail;
} Group;

Group rearrange_group(LinkedList *head) {
  LinkedList *odd_head = head, *even_head = head->next;
  LinkedList *odd_tail = odd_head, *even_tail = even_head;

  for (uint32_t i = 0; i < GROUP_SIZE / 2 - 1; i++) {
    LinkedList *next_odd = even_tail->next;
    LinkedList *next_even = next_odd->next;

    odd_tail->next = next_odd;
    even_tail->next = next_even;

    odd_tail = next_odd;
    even_tail = next_even;
  }

  if (odd_head->value ⩾ 0) {
    odd_tail->next = even_tail->next;
    even_tail->next = odd_head;
    return (Group){even_head, odd_tail};
  } else {
    odd_tail->next = even_head;
    return (Group){odd_head, even_tail};
  }
}
```

```c
int main(int32_t argc, char **argv) {
  srand(clock());

  LinkedList *list = get_input(argc, argv);
  LinkedList_display(list);

  printf("Sorting.\n");

  Group first = rearrange_group(list);
  Group current = first;
  while (current.tail->next != NULL) {
    Group next = rearrange_group(current.tail->next);
    current.tail->next = next.head;
    current = next;
  }

  LinkedList_display(first.head);

  LinkedList_free(first.head);
  return 0;
}
```

Файл 5: main.c

Результати тестування програми

```
>>> ./main -1 2 -3 4 -5 6 -7 8 -9 10
-1 2 -3 4 -5 6 -7 8 -9 10
Sorting.
-1 -3 -5 -7 -9 2 4 6 8 10
```

```
>>> ./main 1 -20 3 -41 5 -62 7 -83 9 -104
1 -20 3 -41 5 -62 7 -83 9 -104
Sorting.
-20 -41 -62 -83 -104 1 3 5 7 9
```

```
>>> ./main
No input received. Generating random sequence...
56 -21 47 -29 78 -24 85 0 97 -26 21 -13 81 -88 72 -84 45 -66 7 -27 40 -94
↪  45 -96 49 -6 58 -60 99 -15 80 -7 89 -28 37 -67 4 -22 19 -2
Sorting.
-21 -29 -24 0 -26 56 47 78 85 97 -13 -88 -84 -66 -27 21 81 72 45 7 -94 -96
↪  -6 -60 -15 40 45 49 58 99 -7 -28 -67 -22 -2 80 89 37 4 19
```

Висновок

Використав однозв'язний список для того щоб динамічно зберегти довільну кількість елементів. Під час виконання програми групував вузли по 10 штук, змінюючи зв'язки між ними всередині цих груп, щоб досягти нового порядку. Врахував правильне звільнення пам'яті після завершення роботи програми (хоча алгоритм не спрацює для будь-якої зацикленості).