Харківський національний університет імені В. Н. Каразіна

ННІ «Комп’ютерних наук та штучного інтелекту»

ЗВІТ

З ЛАБОРАТОРНОЇ РОБОТИ № 1

«Інструментарій програміста»

дисципліна: «Операційні системи»

Виконав: студент групи КC-22

Сікетін Дмитро Сергійович

Перевірив: ст. викладач кафедри ІТММ

Ковальчук Дмитро Миколайович

Харків 2025

**ЗАВДАННЯ №0**

В просторі задано n матеріальних точок. Починаючи з деякого моменту часу точка знайменшою масою зникає та передає свою масу найближчий до ній точці. Цей процес продовжується до тих пір, аж поки не залишиться лише одна точка. Реалізувати цей процес та знайти ту точку, що залишилась.

**Результат виконання завдання:**

**Зображення, що містить текст, знімок екрана, мультимедіа, програмне забезпечення

Вміст, створений ШІ, може бути неправильним.**

Ця програма імітує процес злиття матеріальних точок у просторі. Спочатку користувач вводить, скільки точок буде в моделі, а також їхні параметри (масу й координати). Далі програма виконує наступні кроки:

1. Знаходить точку з найменшою масою.
2. Шукає найближчу до неї точку.
3. Передає масу меншої точки до сусідньої, після чого вона "зникає".
4. Процес повторюється, поки не залишиться тільки одна точка.

У підсумку програма виводить параметри останньої точки.

У коді реалізовано такі речі:

* Структура Point для збереження координат і маси точок.
* Функція find\_closest\_point, яка визначає найближчу точку.
* Функція remove\_point, яка видаляє точку та змінює масив.
* Цикл, що повторює процес об’єднання точок, поки не залишиться одна.
* Робота з пам’яттю (malloc та free) для зберігання точок.

**Висновок:**

Програма добре показує, як точки поступово "зникають", передаючи масу найближчим сусідам. У фіналі лишається одна точка, що має загальну масу всіх початкових точок. Реалізація включає структури, роботу з пам’яттю та алгоритми, тому код вийшов ефективним і зрозумілим.

**ЗАВДАННЯ №4**

Створіть аналог масиву — списку (*ArrayList*) мови *Java*. Реалізуйте наступну функціональність:

1. додавання елемента в кінець списку — метод *add(item);*

2. вставка елемента в середину списку — метод *insert(index, item);*

3. кількість елементів в масиві — метод *size();*

4. видалення елементу по індексу — метод *remove(index);*

5. зміна значення існуючого елементу — метод *set(index, item);*

6. отримання значення заданого елемента — метод *get(index).*

**Робота програми в терміналі:**

Зображення, що містить текст, знімок екрана, програмне забезпечення, Шрифт

Вміст, створений ШІ, може бути неправильним.

**Використання програмою статичної бібліотеки:**

Зображення, що містить текст, знімок екрана, Шрифт

Вміст, створений ШІ, може бути неправильним.

**Використання програмою динамічною бібліотеки:**

Зображення, що містить текст, електроніка, знімок екрана, програмне забезпечення

Вміст, створений ШІ, може бути неправильним.

У цій роботі реалізовано динамічний список ArrayList, який працює як масив змінної довжини. Він підтримує базові операції, такі як додавання (add), вставка (insert), зміна (set), отримання (get), видалення (removeAt) і визначення розміру (size).

Щоб зробити код структурованішим, були створені статична та динамічна бібліотеки, які містять відповідні функції. Проєкт організовано так:

* include — заголовкові файли,
* src — вихідний код,
* lib — бібліотеки,
* obj — об'єктні файли,
* bin — виконувані файли.

Для збірки використовується Makefile, що дозволяє створювати дві версії виконуваного файлу:

1. Зі статичною бібліотекою (lib/librarylist.a).
2. З динамічною бібліотекою (lib/librarylist.dll).

Цікаво, що після компіляції розмір bin/arraylist\_program.exe у обох випадках залишився однаковим — 66073 байтів. Це означає, що в цьому конкретному випадку вибір між статичною та динамічною бібліотекою не вплинув на підсумковий розмір .exe.

Однак у реальних проєктах динамічні бібліотеки можуть зменшити розмір виконуваних файлів, оскільки код бібліотеки зберігається окремо та використовується кількома програмами. Натомість статичні бібліотеки роблять .exe самодостатнім, бо всі необхідні функції вбудовуються в нього.

**Висновок**:

У цій роботі було створено ArrayList, а також дві версії програми: одну зі статичною, іншу з динамічною бібліотекою. Проєкт структурований у кілька директорій для зручності, а збірка автоматизована за допомогою Makefile.

Порівняння показало, що розмір виконуваного файлу залишився незмінним, що, ймовірно, пов’язано з особливостями компіляції або невеликим обсягом бібліотеки. В загальному випадку динамічні бібліотеки спрощують оновлення та дозволяють зменшити розмір .exe, а статичні — роблять програму незалежною від зовнішніх .dll.

**СПИСОК ВИКОРИСТАНИХ КОМАНД ДЛЯ ЗБИРАННЯ ПРОЄКТУ:**

1. Make clean

**Призначення**: Видаляє всі згенеровані об'єкти, виконувані файли та бібліотеки (статичні й динамічні), щоб почати процес збирання з чистої точки.

1. make lib/libarraylist.dll

**Призначення**: Створює динамічну бібліотеку libarraylist.dll, компілюючи файл arraylist.c і формуючи динамічну бібліотеку.

1. Make

**Призначення**: Компільовує весь проект, включаючи об'єктні файли, а також створює виконуваний файл arraylist\_program.exe, використовуючи динамічну бібліотеку.

1. gcc -Wall -Iinclude -c src/arraylist.c -o obj/arraylist.o

**Призначення**: Компілює файл arraylist.c у об'єктний файл arraylist.o, використовуючи вказівку на заголовкові файли через -Iinclude для правильного знаходження шляхів.

1. gcc -Wall -Iinclude -c src/main.c -o obj/main.o

**Призначення**: Компілює файл main.c у об'єктний файл main.o, щоб після цього зібрати виконуваний файл з усіх об'єктних файлів.

1. ar rcs lib/libarraylist.a obj/arraylist.o

**Призначення**: Створює статичну бібліотеку libarraylist.a з об'єктного файлу arraylist.o, який містить реалізацію функцій для роботи з масивами.

1. gcc obj/arraylist.o obj/main.o -Llib -larraylist -o bin/arraylist\_program

**Призначення**: Створює виконуваний файл arraylist\_program.exe, лінкуючи об'єктні файли з бібліотекою libarraylist.a або libarraylist.dll, яка містить необхідні функції.

1. ldd bin/arraylist\_program.exe

**Призначення**: Перевіряє залежності від бібліотек для виконуваного файлу arraylist\_program.exe, надаючи інформацію про те, які динамічні бібліотеки використовуються.

1. ls -l bin/arraylist\_program.exe

**Призначення**: Показує розмір виконуваного файлу arraylist\_program.exe для порівняння результатів після використання різних бібліотек.

1. ls -l lib/libarraylist.a

**Призначення**: Перевіряє розмір статичної бібліотеки libarraylist.a для подальшого порівняння з розмірами інших компонентів.

1. ls -l lib/libarraylist.dll

**Призначення**: Перевіряє розмір динамічної бібліотеки libarraylist.dll для порівняння з іншими файлами проєкту.