

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 КС-22

Сікетін Дмитро Сергійович

Перевірив: ст. викладач кафедри ІТММ

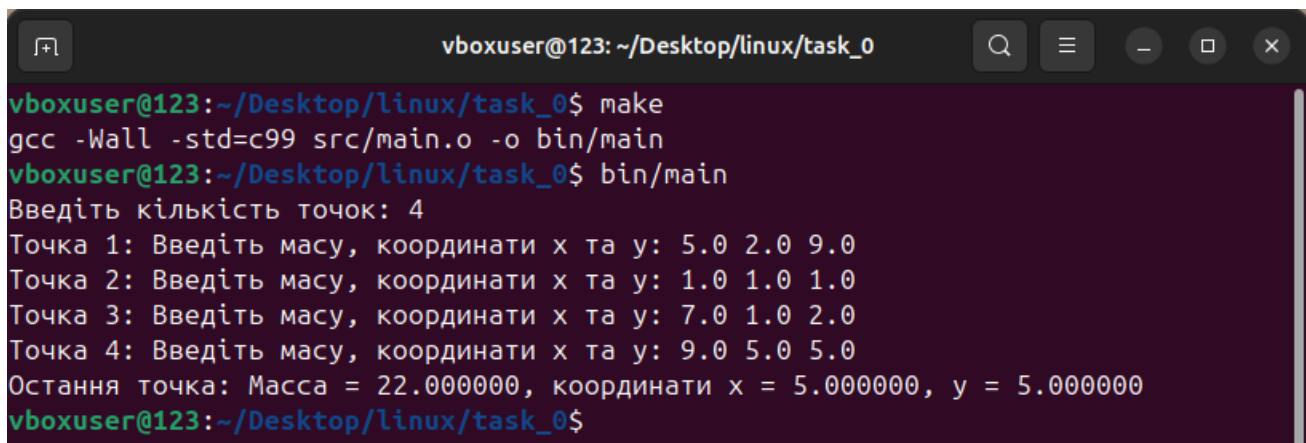
Ковальчук Дмитро Миколайович

Харків 2025

ЗАВДАННЯ №0

В просторі задано n матеріальних точок. Починаючи з деякого моменту часу точка найменшою 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:



```
vboxuser@123: ~/Desktop/linux/task_0
vboxuser@123:~/Desktop/linux/task_0$ make
gcc -Wall -std=c99 src/main.o -o bin/main
vboxuser@123:~/Desktop/linux/task_0$ bin/main
Введіть кількість точок: 4
Точка 1: Введіть масу, координати x та y: 5.0 2.0 9.0
Точка 2: Введіть масу, координати x та y: 1.0 1.0 1.0
Точка 3: Введіть масу, координати x та y: 7.0 1.0 2.0
Точка 4: Введіть масу, координати x та y: 9.0 5.0 5.0
Остання точка: Маса = 22.000000, координати x = 5.000000, y = 5.000000
vboxuser@123:~/Desktop/linux/task_0$
```

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 елемента по індексу — метод *remove(index)*;
5. зміна значення існуючого елемента — метод *set(index, item)*;
6. отримання значення заданого елемента — метод *get(index)*.

Робота програми в терміналі:



```
MINGW64:/d/University/1dima/linux/t4

User@DESKTOP-5007E0P MINGW64 ~
$ cd /d/University/1dima/linux/t4

User@DESKTOP-5007E0P MINGW64 /d/University/1dima/linux/t4
$ make
gcc -Wall -Iinclude -c src/arraylist.c -o obj/arraylist.o
ar rcs lib/libarraylist.a obj/arraylist.o
gcc obj/arraylist.o obj/main.o -Llib -larraylist -o bin/arraylist_program

User@DESKTOP-5007E0P MINGW64 /d/University/1dima/linux/t4
$ bin/arraylist_program
Size of the list: 3
Size of the list after insertion: 4
Value at index 0: 5
Size of the list after removal: 3
```

Використання програмою статичної бібліотеки:

```
User@DESKTOP-5007E0P MINGW64 /d/University/OC/lab1/t4
$ make clean
rm -rf obj/*.o bin/*.exe lib/*.a lib/*.dll

User@DESKTOP-5007E0P MINGW64 /d/University/OC/lab1/t4
$ make
gcc -Wall -Iinclude -c src/arraylist.c -o obj/arraylist.o
gcc -Wall -Iinclude -c src/main.c -o obj/main.o
ar rcs lib/libarraylist.a obj/arraylist.o
gcc obj/arraylist.o obj/main.o -Llib -larraylist -o bin/arraylist_program

User@DESKTOP-5007E0P MINGW64 /d/University/OC/lab1/t4
$ ls -l bin/arraylist_program.exe
-rwxr-xr-x 1 User 197121 66037 Mar  3 19:14 bin/arraylist_program.exe*
```

Використання програмою динамічною бібліотеки:

```
MINGW64:/d/University/1dima/linux/t4
gcc -Wall -Iinclude -c src/arraylist.c -o obj/arraylist.o
gcc -Wall -Iinclude -c src/main.c -o obj/main.o
ar rcs lib/libarraylist.a obj/arraylist.o
gcc obj/arraylist.o obj/main.o -Llib -larraylist -o bin/arraylist_program

User@DESKTOP-5007E0P MINGW64 /d/University/1dima/linux/t4
$ make clean
rm -rf obj/*.o bin/*.exe lib/*.a lib/*.dll

User@DESKTOP-5007E0P MINGW64 /d/University/1dima/linux/t4
$ make lib/libarraylist.dll
gcc -Wall -Iinclude -c src/arraylist.c -o obj/arraylist.o
gcc -shared -o lib/libarraylist.dll obj/arraylist.o

User@DESKTOP-5007E0P MINGW64 /d/University/1dima/linux/t4
$ make
gcc -Wall -Iinclude -c src/main.c -o obj/main.o
ar rcs lib/libarraylist.a obj/arraylist.o
gcc obj/arraylist.o obj/main.o -Llib -larraylist -o bin/arraylist_program

User@DESKTOP-5007E0P MINGW64 /d/University/1dima/linux/t4
$ ls -l bin/arraylist_program.exe
-rwxr-xr-x 1 User 197121 66073 Mar  4 02:46 bin/arraylist_program.exe*
```

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.

2. `make lib/libarraylist.dll`

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

3. `Make`

Призначення: Компілює весь проєкт, включаючи об'єктні файли, а також створює виконуваний файл `arraylist_program.exe`, використовуючи динамічну бібліотеку.

4. `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.

5. `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.

6. `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.

7. `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.

8. `ldd bin/arraylist_program.exe`

Призначення: Перевіряє залежності від бібліотек для виконуваного файлу `arraylist_program.exe`, надаючи інформацію про те, які динамічні бібліотеки використовуються.

9. `ls -l bin/arraylist_program.exe`

Призначення: Показує розмір виконуваного файлу `arraylist_program.exe` для порівняння результатів після використання різних бібліотек.

10.ls -l lib/libarraylist.a

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

11.ls -l lib/libarraylist.dll

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