Державний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Алгоритмізація та програмування»

Тема «Програмування динамічної структури даних – двозв’язний список»

Студента (ки) __1__ курсу AI-211 групи

Спеціальності 122 – «Комп’ютерні науки»

_____Іванов.В.А._____

(прізвище та ініціали)

Керівник ст.викл., к.т.н. Манікаєва О. С.

_____

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____ Оцінка:

ECTS _____

Члени комісії

_____ _____

(підпис) (прізвище та ініціали)

_____ _____

(підпис) (прізвище та ініціали)

_____ _____

(підпис) (прізвище та ініціали)

м. Одеса – 2022 рік

Державний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних систем

ЗАВДАННЯ

НА КУРСОВУ РОБОТУ

студентці Іванову Віталію Андрійовичу група АІ-211

1. Тема роботи

«Програмування динамічної структури даних – двозв’язний список»

2. Термін здачі студентом закінченої роботи 15.06.2022

3. Початкові дані до проекту (роботи)

Варіант 6

Предметна область – вивезення відходів з підприємства.

Реалізувати динамічну структуру даних (двозв’язний список), що містить наступну інформацію: код підприємства, найменування, адреса, телефон, код відходу, найменування, агрегатний стан, дата вивезення, кількість, вартість послуги вивезення. Програма повинна забезпечувати:

− додавання елемента;

− видалення елемента;

− можливість коригування даних;

− виведення всіх даних;

− формування списку підприємств, які вивозили певний вид відходів в задану дату;

− розрахунок вартості наданих послуг з вивезення певного виду відходів з заданого підприємства;

− пошук всіх підприємств, розташованих на заданій вулиці;

− розрахунок загальної кількості вивезених відходів з підприємства за заданий інтервал часу; − сортування по полю кількість, вартість послуги.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити)

Вступ. Теоретичні відомості про двозв’язний список. Програмна реалізація двозв’язного списку. Інструкція користувача. Висновки.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити)

Вступ. Теоретичні відомості про двозв’язний список. Програмна реалізація двозв’язного списку. Інструкція користувача. Висновки. Перелік використаних джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Блок-схема алгоритму – 1 аркуш формату А1.

Завдання видано 28.03.22 _____

(підпис викладача) Завдання

прийнято до виконання 28.03.22 _____

(підпис студента)

АНОТАЦІЯ

В ході курсової роботи розглянуто теоретичні дані про динамічну структуру даних – двозв’язний список, його програмна реалізація, функції для роботи зі списком. Реалізація програми написана за варіантом 6.

Також представлена інструкція користувача, за якою можна з легкістю користуватися програмою. В інструкції написано всі функції, які можна використати в програмі.

В окремому pdf документі запропонована повна графічна схема алгоритму з точним зазначенням обов’язкових креслень.

ABSTRACT

The course work considered theoretical data on the dynamic structure called a double-linked list, its software implementation and functions for working with the list. The implementation of the program is written according to variant 6.

There are also user instructions that allow you to easily use the program. The instructions contain all the features that can be used in the program.

A separate pdf document offers a complete graphical diagram of the algorithm with the exact indication of the required drawings.

ЗМІСТ

ВСТУП

Метою курсової роботи є закріплення і поглиблення знань, одержаних в курсі «Алгоритмізація та програмування», розвиток навичок при виборі представлення початкових даних, вдосконалення техніки використання засобів тестування і налагоджування програми, грамотне оформлення документації на програмну розробку. Темою є програмування динамічної структури даних.

Динамічна структура на відміну від масиву може займати несуміжні ділянки оперативної пам'яті. Динамічні структури широко застосовують і для більш ефективної роботи з даними, розмір яких відомий, особливо для вирішення завдань сортування. Необхідність в динамічних структурах даних зазвичай виникає в наступних випадках:

− використовуються змінні, що мають досить великий розмір (наприклад, масиви великої розмірності) необхідні в одних частинах програми і не потрібні в інших;

− в процесі роботи програми потрібен масив, список або інша структура, розмір якої змінюється в широких межах і важко передбачуваний;

− коли розмір даних, що обробляються в програмі, перевищує обсяг сегмента даних.

Елемент будь-якої динамічної структури складається з двох частин: інформаційної, заради збереження якої і створюється структура, і покажчиків, які забезпечують зв'язок елементів один з одним. Таке уявлення даних в пам’яті називається зв’язковим. Перевагами використання зв’язкового представлення даних є:

− розмір структури обмежується тільки доступним об’ємом машинної пам’яті;

− при зміні логічної послідовності елементів структури потрібно не переміщення даних в пам’яті, а тільки корекція покажчиків;

− велика гнучкість структури.

Недоліками є те, що на поля, які містять вказівники для зв’язування елементів один з одним, збільшується споживання пам’яті та доступ до елементів зв’язної структури може бути менш ефективним за часом.

Видами динамічних структур є лінійні списки, стеки, черги, дерева. Вони розрізняються способами зв'язку окремих елементів і допустимими операціями.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ДВОЗВ’ЯЗНИЙ СПИСОК

1.1 Двозв’язний список

Зв'язаний список – одна з найважливіших структур даних, в якій елементи лінійно впорядковані, але порядок визначається не номерами елементів, а вказівниками, які входять до складу елементів списку та вказують на наступний за даним елемент (в однозв’язних або однобічно зв'язаних списках) або на наступний та попередній елементи (в двозв'язних або двобічно зв'язаних списках). Список має «голову» — перший елемент та «хвіст» — останній елемент.

Зв'язані списки мають серію переваг порівняно з масивами. Зокрема, в них набагато ефективніше виконуються процедури додавання та вилучення елементів. Натомість, масиви набагато кращі в операціях, які потребують безпосереднього доступу до кожного елементу, що у випадку зі зв'язаними списками неможливо та потребує послідовного перебору усіх елементів, які передують даному.

У **двозв'язному списку** кожна структура містить два посилання: на попередню і наступну структури. Таким чином, за списком можна переміщатися від початку до кінця і від кінця до початку. Для доступу до початку і кінця списку повинні бути відомі їх адреси, які можуть зберігатися в глобальних змінних типу покажчик.

В багатьох програмах виникає необхідність організовувати ефективне переміщення по списку як в прямому, так і в зворотному напрямі. Або, ситуації, коли по вказаному елементу, необхідно швидко знайти попередній йому та наступний елементи. У цих ситуаціях можна дати кожному вузлу списку покажчик і на попередній, і на наступний вузол списку, тобто організувати двозв’язний список(рис. 1.1).

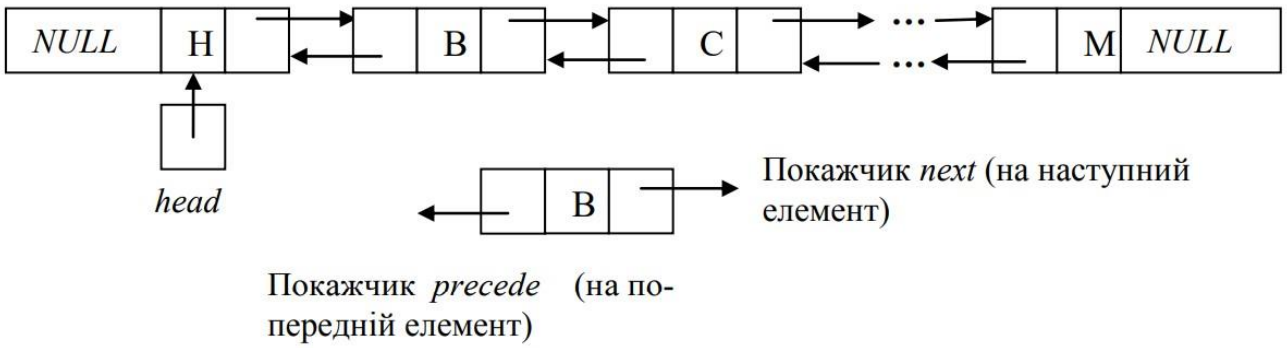

Рис. 1.1 – Схема двозв'язного списку

1.2 Операції зі списками

Додавання елементів

З факту, що список є двічі зв’язаним, не слідкує що досягнути будь-який його елемент можна без обходу списку. Якщо знадобиться додати у впорядкований зв’язаний список новий елемент, для нього необхідно спочатку знайти місце, виділити пам'ять,а потім виконати наступні кроки(рис.1.2):

- Встановити покажчик next нового вузла та наступний;
- Встановити покажчик precede нового вузла на попередній;
- Встановити покажчик precede наступного за новим вузлом, на сам новий вузол;
- Встановити покажчик next попереднього вузла на сам новий вузол.

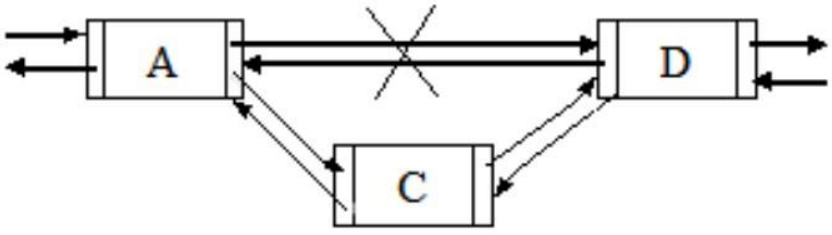

Рис. 1.2 – Схема додавання елементу

Видалення елементів (рис. 1.3)

Наприклад, розглянемо процедуру видалення k-го вузла. Для цього необхідно виконати наступні дії:

- Змінити покажчик на наступний елемент у попереднику k-го вузла так, щоб він посилався на наступний за k-м вузлом;
- Змінити покажчик на попередній елемент в наступному за k-м вузлом так, щоб він посилався на попередній вузол k-го вузла.

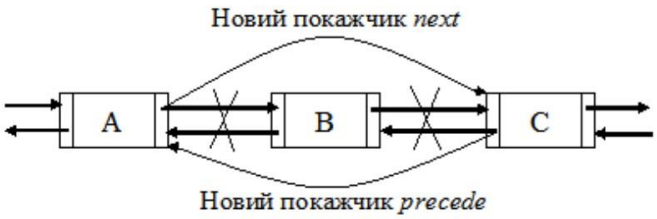

Рис. 1.3 – Схема видалення елементу

При видаленні першого елемента зі списку, другому присвоюється покажчик front. При видаленні останнього елемента, передостанньому присвоюється покажчик на наступний елемент NULL.

2 ПРОГРАМНА РЕАЛІЗАЦІЯ ДВОЗВ’ЯЗНОГО СПИСКУ

2.1 Опис програми

Програма створена для компанії,яка займається вивезенням відходів з підприємств. Програма забезпечує додавання елемента, виведення всіх даних, видалення елемента, можливість коригування даних, формування списку підприємств, які вивозили певний вид відходів в задану дату, розрахунок вартості наданих послуг з вивезення певного виду відходів з заданого підприємства, пошук всіх підприємств, розташованих на заданій вулиці, розрахунок загальної кількості вивезених відходів з підприємства за заданий інтервал часу, сортування по полю кількість, вартість послуги.

Функція int main () в програмі містить в собі виклик всіх доступних користувачеві функцій.

2.2 Меню, введення/виведення даних

Функція menu() виводить пронумерований перелік можливостей програми та приймає вибір користувача у вигляді числа.

Функція create () виділяє пам'ять для структури і працює разом з infoEntering(LEL *p), що дозволяє користувачеві вписати дані.

Функція addElem() дозволяє додати один елемент у кінець списку.

Функція printList(LEL *p) виводить повний список у консоль.. Якщо список є пустим, то у термінал виводяться відповідні коментарі.

Функція fileList() дозволяє записати увесь список у текстовий файл list.txt. А printFile відповідно вивести таблицю даних цього файлу у консоль.

2.3 Видалення даних обраного підприємтва

Функція del() пропонує ввести id підприємства, яке користувач хоче видалити.

2.4 Коригування даних обраного підприємтва

Функція editEl() дає змогу користувачеві заново заповнити дані обраного підприємства. Функція коригування даних працює разом з infoEntering(LEL *p), описаною раніше.

2.5 Виведення списку підприємств за обраною умовою

Функція printList(LEL *p) виводить список, якщо він пустий, то виводить повідомлення про помилку. За допомогою функції printFile() можна вивести у консоль дані з файлу list.txt.

Також у користувача є змога знайти інформацію за датою та вулицею, використовуючи dateSearching() і streetSearching() відповідно.

2.6 Розрахунок загальної кількості вивезених відходів з підприємства за заданий інтервал часу

Функція ttlWeight () дозволяє підрахувати кількість всіх відходів, які були вивезенні за певний проміжок часу. Якщо список є пустим, то у термінал виводяться відповідні коментарі.

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

3.1 Запуск програми

Запустивши програму на консоль виводиться список дій для виконання. Щоб обрати бажану дію достатньо ввести її номер (рис.3.1). Якщо номер невірний, то на консоль буде виведено відповідне повідомлення (рис.3.2).

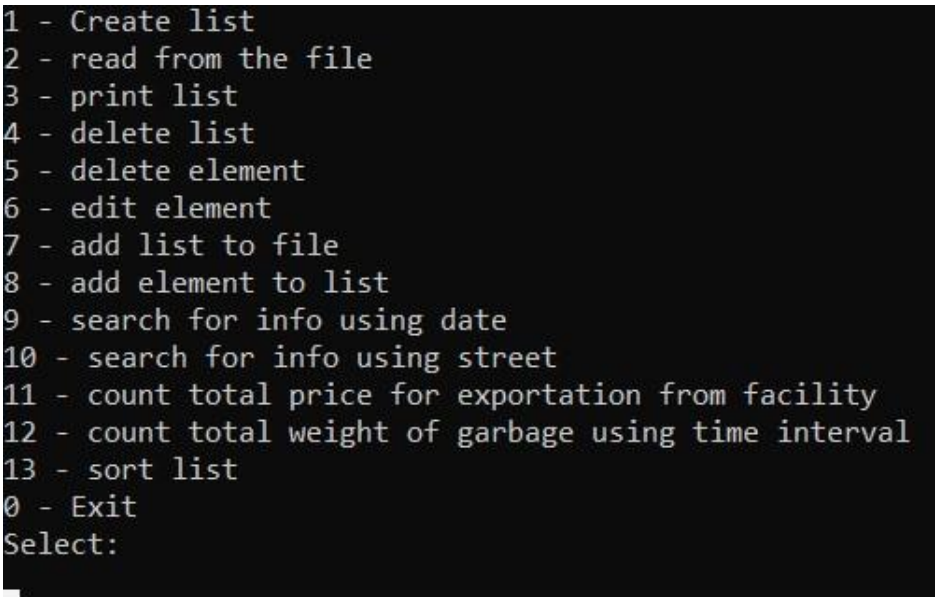

Рис. 3.1 -– Початок і меню програми

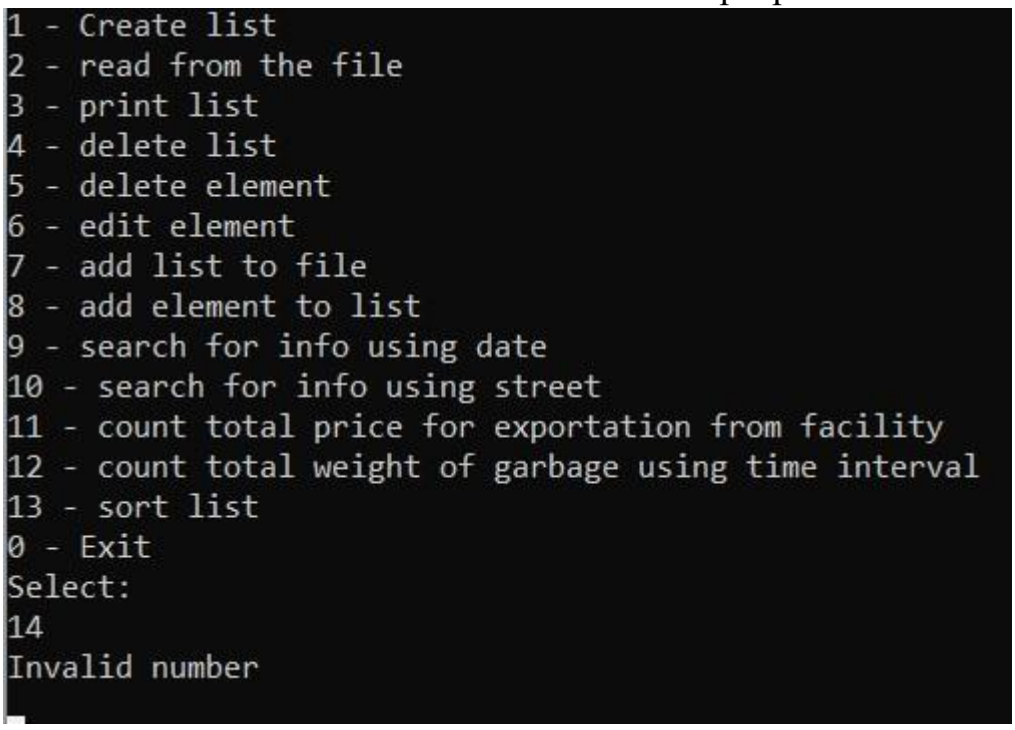

Рис. 3.2 – Повідомлення щодо невірного значення

3.2 Додавання даних

В цій програмі існує 2 варіанти додавання даних:

1)Створення нового списку. Щоб створити список, потрібно вибрати відповідну дію(рис. 3.3). Припинити введення можна, натиснувши клавішу “esc”.

```
1 - Create list
2 - read from the file
3 - print list
4 - delete list
5 - delete element
6 - edit element
7 - add list to file
8 - add element to list
9 - search for info using date
10 - search for info using street
11 - count total price for exportation from facility
12 - count total weight of garbage using time interval
13 - sort list
0 - Exit
Select:
1
Enter id of facility: 1
Enter name of facility: factory
Enter adress of facility: french boul.
Enter contact number of facility: 097 011 1111
Enter date of exportation
Day: 11
Month: 1
Enter code of garbage: 113
Enter name of garbage: glass
Enter state's code of garbage: 1
Enter weight of garbage(kg): 150
Enter price($): 100

Enter id of facility: 2
Enter name of facility: bank
Enter adress of facility: french boul.
Enter contact number of facility: 092 222 2222
Enter date of exportation
Day: 12
Month: 4
Enter code of garbage: 211
Enter name of garbage: leafs
Enter state's code of garbage: 0
Enter weight of garbage(kg): 50
Enter price($): 130
```

Рис. 3.3

Щоб додати дані, потрібно вибрати відповідну дію 8(рис. 3.4). В цьому випадку вас попросять ввести дані для нового елементу, якій буде додано у кінець

списку. Ця функція доступна тільки коли список не порожній, у іншому випадку на консоль буде виведено повідомлення про помилку.

```
2 - read from the file
3 - print list
4 - delete list
5 - delete element
6 - edit element
7 - add list to file
8 - add element to list
9 - search for info using date
10 - search for info using street
11 - count total price for exportation from facility
12 - count total weight of garbage using time interval
13 - sort list
0 - Exit
Select:
8
Enter id of facility: 3
Enter name of facility: school
Enter adress of facility: gagarin str.
Enter contact number of facility: 093 333 3333
Enter date of exportation
Day: 3
Month: 3
Enter code of garbage: 321
Enter name of garbage: farniture
Enter state's code of garbage: 1
Enter weight of garbage(kg): 89
Enter price($): 70
```

Рис. 3.4

3.4 Видалення обраного підприємства

Обираємо необхідний код і пишемо id підприємства, яке хочемо видалити. Перед видаленням програма запитує чи точно її потрібно видалити.

3.5 Коригування даних обраного підприємства

Обираємо необхідний код і пишемо id підприємства, дані якого хочемо відкоригувати (рис. 3.5).

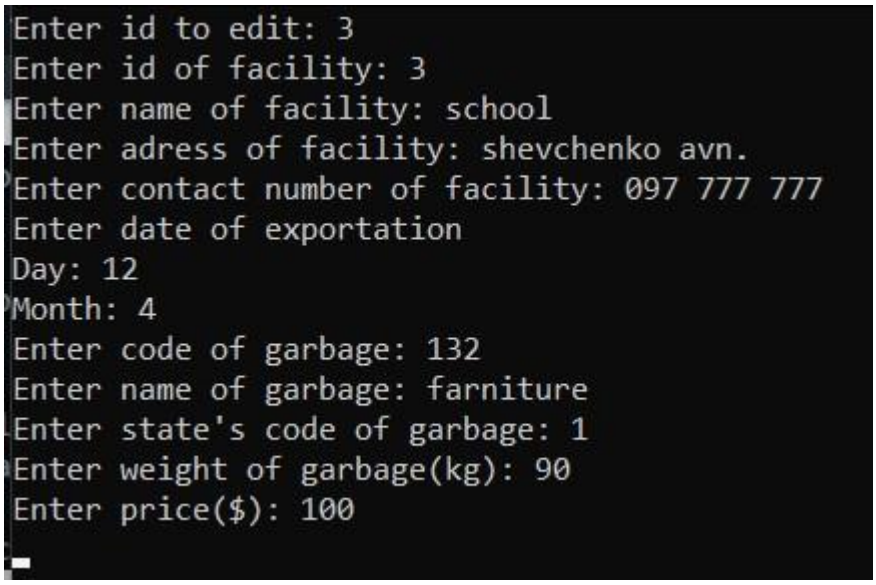

Рис. 3.5 – Коригування даних

3.6 Виведення підприємств за умовою

Функція dateSearching() шукає всю інформацію з заданою датою(рис. 3.6), якщо список пустий – виводить повідомлення про це.

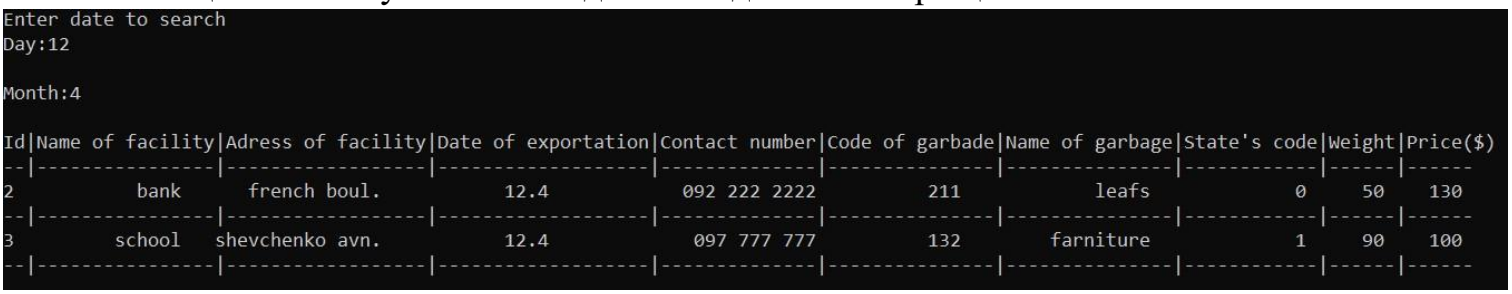

Рис. 3.6- Демонстрація функції dateSearching()

Функція streetSearching() шукає всю інформацію за заданою вулецію(рис. 3.7), якщо список пустий – виводить повідомлення про це.

```
Enter street to search:french boul.

Id|Name of facility|Adress of facility|Date of exportation|Contact number|Code of garbade|Name of garbage|State's code|Weight|Price($)
--|---------------|-----------------|------------------|--------------|---------------|---------------|------------|------|------
1       factory      french boul.          11.1           097 011 1111         113           glass            1    150    100
--|---------------|-----------------|------------------|--------------|---------------|---------------|------------|------|------
2          bank      french boul.          12.4           092 222 2222         211           leafs            0     50    130
--|---------------|-----------------|------------------|--------------|---------------|---------------|------------|------|------
```

Рис. 3.7. - Демонстрація функції streetSearching()

3.7 Розрахунок вартості наданих послуг з вивезення певного виду відходів з заданого підприємства

Обираємо необхідний код, вводимо назву підприємства і програма рахує вартість всіх послуг з вивезення з заданого підприємства(рис. 3.21).

```
Enter name of facility to search:school

Info that we found:
Garbage name:farniture
Garbage code:132
The price it worthed:100

total price:100
```

Рис. 3.8

3.8 Розрахунок загальної кількості вивезених відходів з підприємства за заданий інтервал часу

Обираємо необхідний код, вводимо інтервал часу і програма рахує кількість відходів, яку було вивезено за цей проміжок часу(рис. 3.9).

```
Enter date to begin conting
Day:1
Month:1
Enter date to end conting
Day:12
Month:5

total weight tooked from 1.1 to 12.5:290 kg
```

Рис. 3.9

3.9 Сортування

Викликавши функцію sort(), програма просить обрати критерій за яким вона буде сортувати список(рис. 3.10). Обравши 1, програма сортує список за кількістю відходів(рис. 3.11). В інакшому випадку програма сортує по вартості послуг(рис. 3.12).

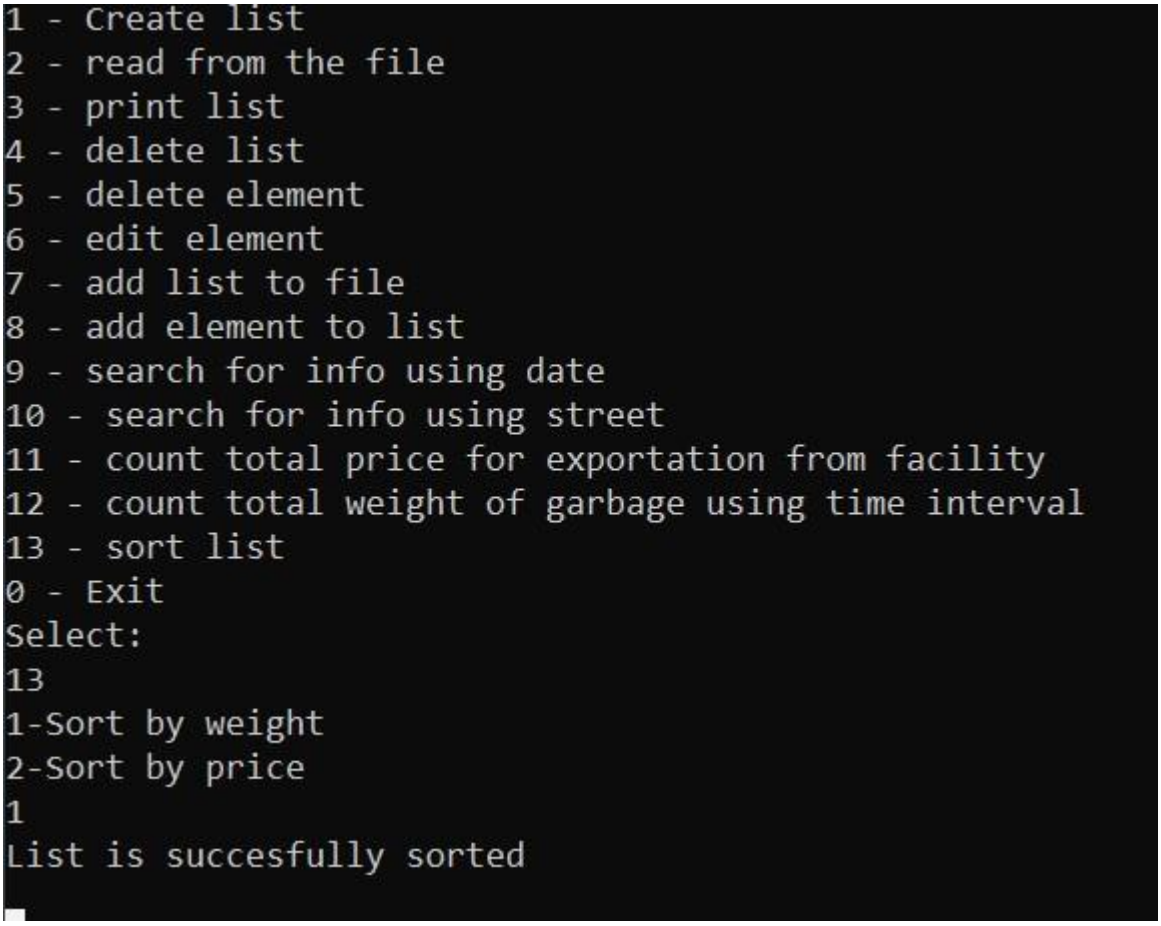

Рис. 3.10

```
1 - Create list
2 - read from the file
3 - print list
4 - delete list
5 - delete element
6 - edit element
7 - add list to file
8 - add element to list
9 - search for info using date
10 - search for info using street
11 - count total price for exportation from facility
12 - count total weight of garbage using time interval
13 - sort list
0 - Exit
Select:
3

Id|Name of facility|Adress of facility|Date of exportation|Contact number|Code of garbade|Name of garbage|State's code|Weight|Price($)
--|----------------|------------------|-------------------|--------------|---------------|---------------|------------|------|------
2          bank      french boul.          12.4            092 222 2222          211           leafs             0     50    130
--|----------------|------------------|-------------------|--------------|---------------|---------------|------------|------|------
3         school   shevchenko avn.          12.4             097 777 777          132        farniture            1     90    100
--|----------------|------------------|-------------------|--------------|---------------|---------------|------------|------|------
1        factory      french boul.          11.1            097 011 1111          113           glass             1    150    100
--|----------------|------------------|-------------------|--------------|---------------|---------------|------------|------|------
```

Рис. 3.11 – Список, сортований за кількістю

```
1 - Create list
2 - read from the file
3 - print list
4 - delete list
5 - delete element
6 - edit element
7 - add list to file
8 - add element to list
9 - search for info using date
10 - search for info using street
11 - count total price for exportation from facility
12 - count total weight of garbage using time interval
13 - sort list
0 - Exit
Select:
3

Id|Name of facility|Adress of facility|Date of exportation|Contact number|Code of garbade|Name of garbage|State's code|Weight|Price($)
--|----------------|------------------|-------------------|--------------|---------------|---------------|------------|------|------
3         school   shevchenko avn.          12.4             097 777 777          132        farniture            1     90    100
--|----------------|------------------|-------------------|--------------|---------------|---------------|------------|------|------
1        factory      french boul.          11.1            097 011 1111          113           glass             1    150    100
--|----------------|------------------|-------------------|--------------|---------------|---------------|------------|------|------
2          bank      french boul.          12.4            092 222 2222          211           leafs             0     50    130
--|----------------|------------------|-------------------|--------------|---------------|---------------|------------|------|------
```

Рис. 3.12 – Список, сортований за вартістю

ВИСНОВКИ

В ході курсової роботи була в повному обсязі написана програма для вивезення відходів з підприємства. Реалізувати динамічну структуру даних (двозв’язний список), що містить наступну інформацію: код підприємства, найменування, адреса, телефон, код відходу, найменування, агрегатний стан, дата вивезення, кількість, вартість послуги вивезення. Програма повинна забезпечувати, додавання елемента, видалення елемента, можливість коригування даних, виведення всіх даних, формування списку підприємств, які вивозили певний вид відходів в задану дату, розрахунок вартості наданих послуг з вивезення певного виду відходів з заданого підприємства, пошук всіх підприємств, розташованих на заданій вулиці. Під час написання програми були покращені теоретичні знання про двозв’язний список та отримані навички програмування з використанням цієї структури даних.

Під час написання програми труднощами були складність знаходження необхідної інформації, відлагодження синтаксичних та семантичних помилок в написаному коді.

Особистий внесок у виконану роботу оцінюється в 90%.

Технічна характеристика розробленої програми: мова програмування C, середовище програмування – CodeBlocks, версія 20.3, обсяг коду – 816 рядків.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шпак З.Я. Програмування мовою С / З.Я. Шпак. – Л.: Оріяна-Нова, 2006. – 432 с.

2. Глибовець А.М. Практикум з мови програмування Сі / А.М. Глибовець, М.М. Глибовець, В.С. Проценко. – К.: Вид. дім «Київо-Могил. академии», 2010. – 209 с.

3. Бабілунга О.Ю. Конспект лекцій з дисципліни "Алгоритмізація та програмування" для спеціальності 122 Комп'ютерні науки, Одеса, 2021. URL: el.opu.ua/course/.

4. Методичні вказівки до курсової роботи з дисципліни «Алгоритмізація та програмування» для студентів спеціальності 122 – «Комп’ютерні науки» /Укл.: О.Ю. Бабілунга. – Одеса: НУ «Одеська політехніка», 2021. – 19 с.

ДОДАТОК А КОД ПРОГРАМИ

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <string.h>
#include <windows.h>
#define MAX_LEN 100

typedef struct facility
   {
   int id;
   char name[MAX_LEN];     char
adress[MAX_LEN];     char
phone[MAX_LEN];     int
garbageCode;     char
garbageName[MAX_LEN];     int
stateCode;//0-жидкий 1-твердый     int
weight;
   //char date[MAX_LEN];
   int day,month;
   int price;

   }INFO;

typedef struct list_elem
   {
   INFO data;     struct list_elem *prev;// покажчик на
попередню структуру     struct list_elem *next;// покажчик на
наступну структуру

   }LEL;

   LEL *list_beg,*list_end;

   int kod;

   void create(void); // створення
void printList(LEL *); // перегляд
   void del(void); // видалення
void printFile();
void del(void);
void menu();
void addElem();
void infoEntering(LEL *p);
void editEl();
```

```
void fileList();
void delette();
void dateSearching();
void ttlPrice();
void streetSearching();
void sort(LEL *start);
void swapData(LEL *a,LEL *tmp);
void ttlWeight();

   FILE *f;

int main() {
SetConsoleCP(1251); //встановлення кодування Windows-1251 в потік введення
   SetConsoleOutputCP(1251); // встановлення кодування Windows-1251 в потік виведення
system("cls");

   menu();

   free(list_beg);

   return 0;
}

void create(void)
{
   LEL *p,*pred;
   pred=NULL;
   do
   {
      p=(LEL *)malloc(sizeof(LEL));
      infoEntering(p);        p-
>prev=pred;
   if (pred!=NULL)
   {
      pred->next=p;
   }    else
list_beg=p;
pred=p;
   puts("");

   }while (getch()!=27);
list_end=p;
   list_end->next=NULL;
}
void printList(LEL *p)
{
```

```
   if(list_beg==NULL)
   {
    puts("There is no list to print");
    getch();
   }

else
   {
   if(p==list_beg)
   {
       printf("\nId|Name of facility|Adress of facility|Date of exportation|Contact number|Code of
garbade|Name of garbage|State's code|Weight|Price($)");
        printf("\n--|----------------|-----------------|------------------|-------------|--------------|-----------
----|------------|------|------");
       while (p != NULL)
   {
       printf("\n%d%15s%18s%13d.%d%24s%13d%17s%14d%7d%7d",p->data.id,p-
>data.name,p->data.adress,p->data.day,p->data.month,p->data.phone,p->data.garbageCode,p-
>data.garbageName,p->data.stateCode,p->data.weight,p->data.price);
       printf("\n--|----------------|-----------------|------------------|-------------|--------------|-----------
--|------------|------|------");

       p=p->next;
   }
}
else
{
       puts("False adress");
   }
getch();
}
}

void printFile()
{
       char ch;
       f=fopen("list.txt","r");
while(!feof(f))
       {
       ch = getc(f);
       putchar(ch);
       }
getch();
fclose(f);
}
```

```
void del(void)
{
   LEL *p,*temp;
   int del;
system("cls");
   printf("Enter id to delete: ");
   scanf("%d",&del);
   p=list_beg;

   while(p!=NULL)
   {

   if (del==p->data.id)
   {
      if(p==list_beg)
      {
         list_beg=p->next;
         list_beg->prev=NULL;
         free(p);
p=list_beg;        }
      else if(p==list_end)
      {
         list_end=p->prev;
         list_end->next=NULL;
         free(p);
p=list_end;
      }
else
      {
         p->next->prev=p->prev;           p->prev-
>next=p->next;
         temp=p;
p=p->next;
free(temp);
      }
}
else
      p=p->next;
   }

}

void menu() {         system("cls");  puts("1 - Create list");
puts("2 - read from the file");  puts("3 - print list");  puts("4 -
delete list");  puts("5 - delete element");  puts("6 - edit
element");  puts("7 - add list to file");  puts("8 - add element
to list");  puts("9 - search for info using date");  puts("10 -
```

```
search for info using street");  puts("11 - count total price for
exportation from facility");  puts("12 - count total weight of
garbage using time interval");  puts("13 - sort list");  puts("0 -
Exit");  // Вибір бажаної дії  puts("Select:");  scanf("%d",
&kod);
 // Запуск функції згідно обраної дії
switch(kod)
 {  case
1:
    create(); break;
case 2:
    //printList(list_beg);
    printFile();      break;
case 3:
printList(list_beg);
break;  case 4:
delette();      break;  case
5:      del();      break;
case 6:      editEl();
break;  case 7:
fileList();      break;  case
8:      addElem();
break;  case 9:
dateSearching();     break;
case 10:
streetSearching();
break;  case 11:
ttlPrice();     break;  case
12:     ttlWeight();
break;  case 13:
sort(list_beg);     break;
case 0:      exit(1);
break;  default:
puts("Invalid number");
getch();     break;  }
if(kod!=0)  {      menu();
} } void addElem()
{
   if(list_beg==NULL)
   {
system("cls");
       puts("You have to create list before using this function");
getch();     }     else     {
   LEL *p;
   p=(LEL *)malloc(sizeof(LEL));
p->prev=list_end;     list_end->next=p;
   p->next=NULL;
```

```
   infoEntering(p);
   list_end=p;
   } }
void infoEntering(LEL *p)
{         printf("Enter id of facility:
");         scanf("%d",&p->data.id);
        fflush(stdin);
        printf("Enter name of facility: ");
gets(p->data.name);         fflush(stdin);
        printf("Enter adress of facility: ");
gets(p->data.adress);         fflush(stdin);
        printf("Enter contact number of facility: ");
gets(p->data.phone);         fflush(stdin);
printf("Enter date of exportation\nDay: ");
scanf("%d",&p->data.day);
        fflush(stdin);
printf("Month: ");
scanf("%d",&p->data.month);
fflush(stdin);
        printf("Enter code of garbage: ");
        scanf("%d",&p->data.garbageCode);
fflush(stdin);
        printf("Enter name of garbage: ");
        gets(p->data.garbageName);
        fflush(stdin);         printf("Enter state's
code of garbage: ");
        scanf("%d",&p->data.stateCode);
fflush(stdin);
        printf("Enter weight of garbage(kg): ");
scanf("%d",&p->data.weight); fflush(stdin);
        printf("Enter price($): ");         scanf("%d",&p-
>data.price);
}
void editEl() {
   LEL *p;     int
ed;
system("cls");
   p=list_beg;

   if(p==NULL)
   {
    puts("No elements to edit");
    getch();
   }
```

```
else
{
      printf("Enter id to edit: ");
scanf("%d",&ed);
    while(p!=NULL)
   {
      if(ed==p->data.id)
      {
         infoEntering(p);

      getch();
      }

      p=p->next;
   }
   } }
void fileList()
{
      LEL *p=list_beg;          f=fopen("list.txt","w");      fprintf(f,"Id|Name of facility|Adress of facility|Date of exportation|Contact number|Code of garbade|Name of garbage|State's code|Weight|Price($)");
   fprintf(f,"\n--|----------------|------------------|------------------|--------------|---------------|-------------|------------|------|------");
      while (p != NULL)
   {
      fprintf(f,"\n%d%15s%18s%13d.%d%24s%13d%17s%14d%7d%7d",p->data.id,p->data.name,p->data.adress,p->data.day,p->data.month,p->data.phone,p->data.garbageCode,p->data.garbageName,p->data.stateCode,p->data.weight,p->data.price);
      fprintf(f,"\n--|----------------|------------------|------------------|--------------|---------------|-------------|------------|------|------");

      p=p->next;
   }
fclose(f);
}
void delette()
{
   LEL *p=list_beg;
   if(list_beg!=0)
   {
   list_beg=NULL;
   }
else    {
      puts("list is already deleted");
      getch();
```

```
   } }
void dateSearching()
{
   LEL *p=list_beg;
   if(p==NULL)
   {
      system("cls");
      puts("You have to create list before using this function");
   }
else
   {
      //char srchDate[MAX_LEN];
int srchDay,srchMonth;
      system("cls");
      printf("Enter date to search\nDay:");
      fflush(stdin);
//gets(&srchDate);
scanf("%d",&srchDay);
printf("\nMonth:");
scanf("%d",&srchMonth);
         printf("\nId|Name of facility|Adress of facility|Date of exportation|Contact number|Code
of garbade|Name of garbage|State's code|Weight|Price($)");
       printf("\n--|----------------|-----------------|------------------|--------------|--------------|-----------
----|------------|------|------");
      while (p != NULL)
   {
      //if(strcmp(srchDate,p->data.date)==0)
      if(srchDay==p->data.day && srchMonth==p->data.month)
      {
      printf("\n%d%15s%18s%13d.%d%24s%13d%17s%14d%7d%7d",p->data.id,p-
>data.name,p->data.adress,p->data.day,p->data.month,p->data.phone,p->data.garbageCode,p-
>data.garbageName,p->data.stateCode,p->data.weight,p->data.price);
      printf("\n--|----------------|-----------------|------------------|--------------|--------------|-----------
--|------------|------|------"); }
      p=p->next;
   }    }
getch(); }
void ttlPrice()
{
   LEL *p=list_beg;
   if(p==NULL)
   {
system("cls");
      puts("You have to create list before using this function");
getch();    }    else    {        int ttlPrice=0;
      char srchName[MAX_LEN];
```

```
        system("cls");
fflush(stdin);
        printf("Enter name of facility to search:");
        gets(&srchName);
printf("\nInfo that we found:");
        while (p != NULL)
    {
        if(strcmp(srchName,p->data.name)==0)
        {
        printf("\nGarbage name:%s",p->data.garbageName);
printf("\nGarbage code:%d",p->data.garbageCode);         printf("\nThe
price it worthed:%d\n",p->data.price);         ttlPrice=ttlPrice+p-
>data.price;

        }
        p=p->next;
    }
    printf("\ntotal price:%d",ttlPrice);
    getch();
}
}

void streetSearching()
{
    LEL *p=list_beg;
    if(p==NULL)
    {
system("cls");
        puts("You have to create list before using this function");
    }
else     {
char
srchstre
et[MA
X_LEN
];
system(
"cls");
        printf("Enter street to search:");
        fflush(stdin);
gets(&srchstreet);
//scanf("%s",&srchDate);
          printf("\nId|Name of facility|Adress of facility|Date of exportation|Contact number|Code
of garbade|Name of garbage|State's code|Weight|Price($)");
         printf("\n--|---------------|-----------------|------------------|-------------|--------------|-----------
----|------------|------|------");
        while (p != NULL)
```

```
   {
      if(strcmp(srchstreet,p->data.adress)==0)
      {
      printf("\n%d%15s%18s%13d.%d%24s%13d%17s%14d%7d%7d",p->data.id,p-
>data.name,p->data.adress,p->data.day,p->data.month,p->data.phone,p->data.garbageCode,p-
>data.garbageName,p->data.stateCode,p->data.weight,p->data.price);
      printf("\n--|----------------|-----------------|------------------|-------------|--------------|-----------
--|------------|------|------");
      }
      p=p->next;
   }    }
getch();
}

void sort(LEL *start)
{
   if(list_beg==NULL)
   {
      system("cls");
      puts("There is nothing to sort");
   }
else
   {
   LEL *a;
LEL *tmp;
int t;    int
flag=1;
   puts("1-Sort by weight\n2-Sort by price");
scanf("%d",&t);    if(t==1)
   while(flag==1)
   {
      tmp=start;
a=tmp->next;        flag=0;
      while(a)
      {
         if((tmp->data.weight)>(a->data.weight))
         {
            swapData(a,tmp);
flag=1;
         }
         tmp=tmp->next;
         a=a->next;
      }

   }
```

```
   else
while(flag==1)
   {
tmp=start;
a=tmp->next;
flag=0;
while(a)
      {
         if((tmp->data.price)>(a->data.price))
         {
            swapData(a,tmp);
flag=1;
         }
         tmp=tmp->next;
         a=a->next;
      }

   }
   puts("List is succesfully sorted");
   getch();
   }

}
void swapData(LEL *a,LEL *tmp)
{
   int tInt=0;
   char tStr[MAX_LEN];

      tInt=tmp->data.weight;        tmp->data.weight=a-
>data.weight;        a->data.weight=tInt;

      tInt=tmp->data.month;        tmp->data.month=a-
>data.month;
      a->data.month=tInt;

      tInt=tmp->data.garbageCode;        tmp->data.garbageCode=a-
>data.garbageCode;
      a->data.garbageCode=tInt;

      tInt=tmp->data.id;
      tmp->data.id=a->data.id;
      a->data.id=tInt;

      tInt=tmp->data.price;        tmp->data.price=a-
>data.price;        a->data.price=tInt;
```

```
        tInt=tmp->data.stateCode;         tmp->data.stateCode=a-
>data.stateCode;
        a->data.stateCode=tInt;

        tInt=tmp->data.day;         tmp->data.day=a-
>data.day;
        a->data.day=tInt;

        strcpy(tStr,tmp->data.adress);         strcpy(tmp->data.adress,a-
>data.adress);
        strcpy(a->data.adress,tStr);

        strcpy(tStr,tmp->data.garbageName);         strcpy(tmp->data.garbageName,a-
>data.garbageName);
        strcpy(a->data.garbageName,tStr);

        strcpy(tStr,tmp->data.name);         strcpy(tmp->data.name,a-
>data.name);         strcpy(a->data.name,tStr);

        strcpy(tStr,tmp->data.phone);         strcpy(tmp->data.phone,a-
>data.phone);         strcpy(a->data.phone,tStr);

} void
ttlWeight()
{
  LEL *p=list_beg;
   if(p==NULL)
   {
system("cls");
        puts("You have to create list before using this function");
getch();     }     else     {          int ttlweight=0;          int
fromDay,fromMonth;          int toDay,toMonth;
system("cls");          fflush(stdin);
        printf("Enter date to begin conting \nDay:");
scanf("%d",&fromDay);
        printf("Month:");
scanf("%d",&fromMonth);

        printf("Enter date to end conting \nDay:");
scanf("%d",&toDay);         printf("Month:");
scanf("%d",&toMonth);
        while (p != NULL)
   {
        if((p->data.month>fromMonth && p->data.month<toMonth) || (p>data.month==fromMonth
&& p->data.day>fromDay) || (p->data.month==toMonth && p>data.day<toDay))
        {
        ttlweight=ttlweight+p->data.weight;
```

```
        }
        p=p->next;
    }
    printf("\ntotal weight tooked from %d.%d to %d.%d:%d
kg",fromDay,fromMonth,toDay,toMonth,ttlweight); getch();
}
}
```