

КОНТЕЙНЕРИ

Лабораторна робота №6

Мета:

- Отримати навички розробки власних контейнерів на базі уснуючих класів.

1 ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Визначити клас колекції `CapacityList` для зберігання даних згідно своєї тематичної області. Перевантажити оператор `[]`, та реалізувати можливість зберігання даних у файл та подальше завантаження із файлу, базуючись на `FileStoreLibrary`

2 РОЗРОБКА ПРОГРАМИ

Для реалізації програми було створено структуру, що являє собою звено списку та клас, що реалізує двухзв'язний список.

2.1 Засоби ООП

У розробленій програмі використані наступні засоби ООП:

- розділення програми на ієрархію класів (інкапсуляція);
- поліморфізм;
- спадкування;
- абстракція (віртуальність);

2.2 Ієрархія та структура класів

На рис.2.2 наведена ієрархія розроблених класів

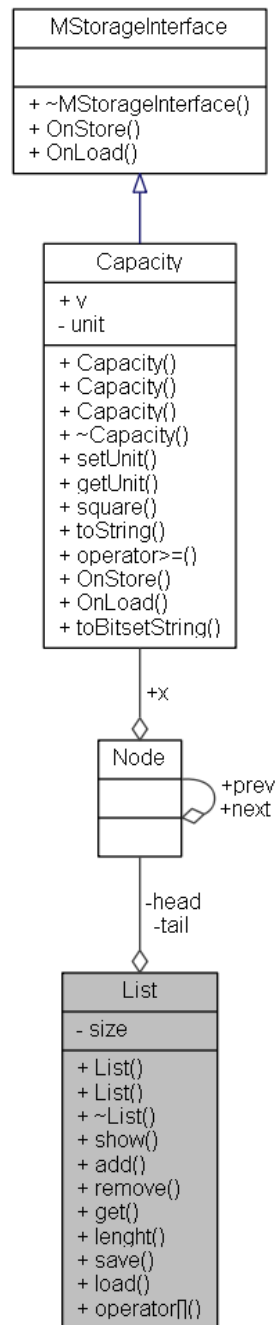


Рисунок 2.2 – Ієрархія класів

2.3 Опис програми

На рис.2.3 наведена структура розробленого проекту

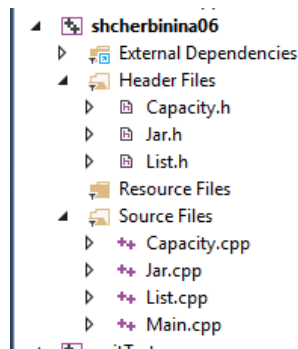


Рисунок 2.3 – Структура проекту

Призначення спроектованих класів наведено на рис.2.4

Класи, структури, об'єднання та інтерфейси з коротким описом.

C Capacity	Клас, що містить реалізацію ємності
C CFileStorage	
C Jar	Клас, що містить реалізацію банки
C List	Класс реализующий двухсвязный список для объектов типа Capacity
C MStorageInterface	
C Node	Структура являющаяся звеном списка

Рисунок 2.4 – Призначення класів

2.4 Важливі фрагменти програми

Перевантажений оператор [] та функція get(int pos):

```
Capacity* List::get(int pos) {
    if (pos<0 || pos>(size-1)){
        cout << "Illegal index\n";
        return (new Capacity);
    }
    Node *tmp = head;
    for (int i = 0; i < pos; i++)
        tmp = tmp->next;
    return &(tmp->x);
}

Capacity* operator[](int pos){
    return get(pos);
}
```

Функція запису у файл:

```
void List::save(string filename)
{
    ofstream fileStream;
    fileStream.open(filename, ios_base::out | ios_base::binary | ios_base::trunc);
    for (int i = 0; i < this->size; i++){
        get(i)->OnStore(fileStream);
        fileStream << "\n";
    }
    fileStream.close();
}
```

Функція зчитування з файлу:

```
void List::load(string filename, int size)
{
    ifstream fileStream;
    fileStream.open(filename, ios_base::in | ios_base::binary);
    for(int i = 0 ; i<size;i++){
        Capacity temp;
        temp.OnLoad(fileStream);
        this->add(temp);
    }
    fileStream.close();
}
```

Функція main():

```
/**
 * Точка входу в програму
 */
int main() {
    setlocale(LC_ALL, "Russian");

    Capacity test1(1, 1000);
    Capacity test2(ml, 500);
    Capacity test3(cubicMeter, 2000);

    List list;
    list.add(test1);
    list.add(test2);
    list.add(test3);

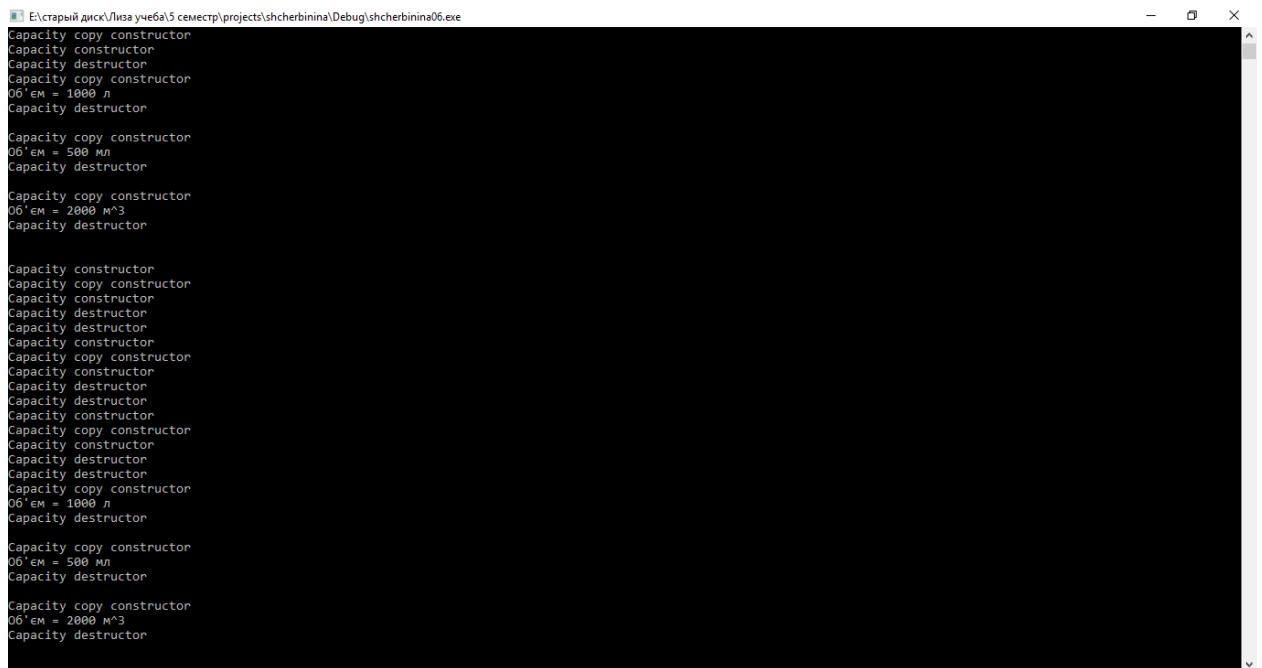
    list.show();

    list.save("temp.bin");

    List list2;
    list2.load("temp.bin", 3);
    list2.show();

    getch();
    return 0;
}
```

3 РЕЗУЛЬТАТИ РОБОТИ



```
E:\Старый диск\Лиза учеба\5 семестр\projects\shcherbinina\Debug\shcherbinina06.exe
Capacity copy constructor
Capacity constructor
Capacity destructor
Capacity copy constructor
06'em = 1000 л
Capacity destructor

Capacity copy constructor
06'em = 500 мл
Capacity destructor

Capacity copy constructor
06'em = 2000 м^3
Capacity destructor

Capacity constructor
Capacity copy constructor
Capacity constructor
Capacity destructor
Capacity destructor
Capacity constructor
Capacity copy constructor
Capacity constructor
Capacity destructor
Capacity destructor
Capacity constructor
Capacity copy constructor
Capacity constructor
Capacity destructor
Capacity destructor
Capacity copy constructor
06'em = 1000 л
Capacity destructor

Capacity copy constructor
06'em = 500 мл
Capacity destructor

Capacity copy constructor
06'em = 2000 м^3
Capacity destructor
```

Рисунок 3.1 – Приклад роботи програми

ВИСНОВКИ

В результаті лабораторної роботи було отримано навички розробки програм з власними контейнерами.