ЛАБОРАТОРНА РОБОТА №2

Тема: Робота з лінійними списками. Конструктор і деструктор класу
Мета: Навчитись використовувати конструктори і деструктори класів, створювати класи для опису лінійних списків

Завдання:
1 Реалізувати клас згідно варіанту індивідуального завдання, що містить закриті дані, а саме два типа даних: числове значення та символьний рядок, який реалізований через вказівник на char (char *). Для спрощення задачі можна замінити char * на тип даних string (на оцінку «задовільно»).
2 Реалізувати методи: конструктори, деструктор, input() – метод для запит у користувача даних та їх зчитування з клавіатури у поля класу, print() – константний метод виводу даних на екран, методи доступу до закритих даних.
3 У функції main() створити декілька екземплярів класу статично і динамічно (із введенням даних із клавіатури користувачем). У звіті продемонструвати дію всіх конструкторів і методів за допомогою знімків екрану.
4 *Реалізувати клас однозв'язного списку List, який міститиме об'єкти класу, розробленого згідно варіанту індивідуального завдання. Продемонструвати роботу списку, добавивши декілька елементів, після чого вивести на екран увесь список.

Варіант 10.
class Furniture {
char *room;
int weight;
public:
Furniture();
Furniture(char * room, int weight);
Furniture(const Furniture&);
void setRoom(char * room);
char * getRoom( );
void setWeight(int weight);
int getWeight( );
void print() const;
void input ();
~Furniture(); };

Код програми:

Common.h:

```
#pragma once

#include <iostream>
#include <string.h>
#define N 16
```

Furniture.h:

```
#pragma once

class Furniture {
private:
    char* room;
    int weight;
public:
    Furniture();
    Furniture(char* room, int weight);
    Furniture(const Furniture&);
    void setRoom(char* room);
    char* getRoom() const;
    void setWeight(int weight);
    int getWeight() const;
    void print() const;
    void input();
    ~Furniture();
};
```

Furniture.cpp:

```
#include "Furniture.h"
#include "Common.h"
using namespace std;

Furniture::Furniture() {
    cout << "Basic furniture constructor" << endl;
    this->weight = 0;
    room = nullptr;
}
Furniture::Furniture(char* room, int weight) {
    cout << "Parametrized furniture constructor" << endl;
    setRoom(room);
    setWeight(weight);
}
Furniture::Furniture(const Furniture& src) {
    cout << "Parametrized furniture constructor (link)" << endl;
    setRoom(src.getRoom());
    setWeight(src.getWeight());
}
Furniture::~Furniture() {
    cout << "Furniture dectructor" << endl;
    if (room) {
        delete[] room;
    }
}

void Furniture::setRoom(char* room) {
    if (this->room) {
        delete[] this->room;
    }
    int roomLen = strlen(room) + 1;
    this->room = new char[roomLen];
    strcpy_s(this->room, roomLen, room);
}

void Furniture::setWeight(int weight) {
    this->weight = weight;
}

char* Furniture::getRoom() const {
    return this->room;
```

```
}

int Furniture::getWeight() const {
    return this->weight;
}

void Furniture::input() {
    char nroom[N];
    int nweight;
    cout << "Enter room: ";
    cin >> nroom;
    setRoom(nroom);
    cout << "Enter weight of furniture: ";
    cin >> nweight;
    setWeight(nweight);
}

void Furniture::print() const {
    cout << "The furniture for room \"" << getRoom() << "\" has weight " <<
getWeight() << endl;
}
```

List.h:

```
#pragma once
#include "Furniture.h"

class List {
private:
    List* next;
    Furniture data;
public:
    List(const Furniture& newData, List* oldList = nullptr);
    ~List();
    void printList();
};
```

List.cpp:

```
#include "List.h"
#include "Common.h"
using namespace std;

List::List(const Furniture& newData, List* oldList) {
    cout << "List parametrized constructor" << endl;
    data.setRoom(newData.getRoom());
    data.setWeight(newData.getWeight());
    next = oldList;
}
List::~List() {
    cout << "List destructor" << endl;
    if (next) {
        delete next;
    }
}
void List::printList() {
    data.print();
    if (next)
        next->printList();
}
```

lab2.cpp:

```cpp
#include "List.h"
#include "Furniture.h"
#include "Common.h"
using namespace std;

int main()
{
    Furniture* dynFurniture = new Furniture;
    char roomBuffer[N];

    strcpy_s(roomBuffer, N, "kitchen");
    dynFurniture->setRoom(roomBuffer);
    dynFurniture->setWeight(19);

    Furniture furniture1;
    furniture1.input();

     Furniture furniture3(*dynFurniture);
     furniture3.setWeight(55);

     cout << "Enter room: ";
     cin >> roomBuffer;
     Furniture furniture2(roomBuffer, 33);

     dynFurniture->print();
     furniture2.print();
     furniture3.print();

     delete dynFurniture;

     List* myList = new List(furniture3);
     myList = new List(furniture2, myList);
     myList->printList();
     delete myList;

    return 0;
}
```

Результат:

```
Basic furniture constructor
Basic furniture constructor
Enter room: bath
Enter weight of furniture: 12
Parametrized furniture constructor (link)
Enter room: living_room
Parametrized furniture constructor
The furniture for room "kitchen" has weight 19
The furniture for room "living_room" has weight 33
The furniture for room "kitchen" has weight 55
Furniture dectructor
Basic furniture constructor
List parametrized constructor
Basic furniture constructor
List parametrized constructor
The furniture for room "living_room" has weight 33
The furniture for room "kitchen" has weight 55
List destructor
List destructor
Furniture dectructor
Furniture dectructor
Furniture dectructor
Furniture dectructor
Furniture dectructor
```

Висновок: навчився використовувати конструктори і деструктори класів, створювати класи для опису лінійних списків.