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:

типа даних: числове значення та рядок, реалізований через вказівник на char (char *).

2 Реалізувати методи:

 конструктор по замовчуванню;

 конструктор ініціалізації клас(char*, int);

 конструктор копіювання клас(const клас&);

 деструктор;

 Input() – 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), продемонструвати дію всіх конструкторів і методів.

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 7. class Airplane

{ char *Model; int Power;

public: Airplane();

Airplane(char * , int);

Airplane(const Airplane&);

void SetModel(char *);

```
char * GetModel( );
void SetPower( int );
 int GetPower( );
void Print() const;
void Input ();
~Airplane(); };
```

Код програми:

```
#include<iostream>

template <typename T>
class Airplane
{
private:
   template<typename T>
   class Node // шаблонний клас нод однозвязний список
   {
   public:
      Node* pNext;
      T info;
      Node(T info = T(), Node* pNext = nullptr)
      {
         this->info = info;
         this->pNext = pNext;
      }
   };
   Node<T> *m_model;
   int m_power;
```

```
public:
    Airplane();
    Airplane(T* m_model, int m_power);
    Airplane(const Airplane& obj);
    void setModel(const T* m_model, int lenght);//додає масив в кінець
    T* getModel();
    void setPower(int); //присвоює потужність
    int getPower();//виводить потужність
    void print(); //виводить всю інформацію
    void input(); // ввід обєкта
    void pop_front (); // видаляє голову
    void pushBack (T); // добавляє елемент в кінець
    T getModel (int index); // виводить одну модель
    ~Airplane();
};
  template<typename T>
Airplane<T>::Airplane ():m_power(0), m_model(nullptr){}

template<typename T>
Airplane<T>::Airplane (T *model, int power){
     setModel(model,power);
}

template<typename T>
Airplane<T>::Airplane (const Airplane &obj) { m_model = obj.m_model;
m_power = obj.m_power; }
```

```
template<typename T>
void Airplane<T>::setModel (const T *model, int lenght)
{
   for( int i = 0; i < lenght; i++ ) {
      pushBack (model[i]);
   }
}

template<typename T>
T *Airplane<T>::getModel ()
{
   return *m_model.info;
}


template<typename T>
void Airplane<T>::setPower (int a)
{
   m_power = a;
}

template<typename T>
int Airplane<T>::getPower ()
{
   return m_power;
}
```

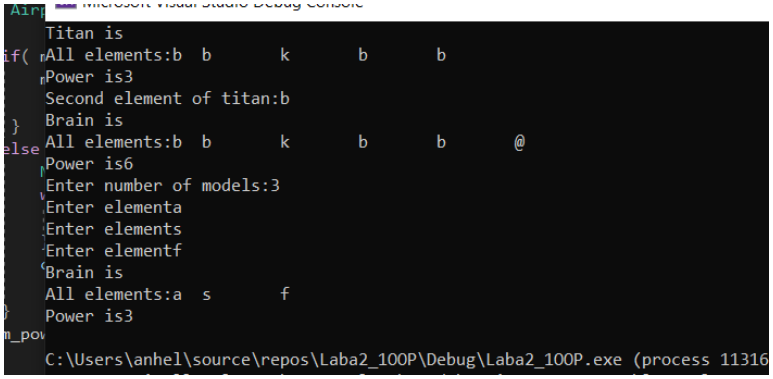
```
template<typename T>
void Airplane<T>::print ()
{
    Node<T> *curr = this->m_model;
    std::cout << "\nAll elements:";
    while( curr != nullptr ) {
        std::cout << curr->info << "\t";
        curr = curr->pNext;

    }
    std::cout << std::endl;
    std::cout << "Power is" << m_power << std::endl;
}

template<typename T>
void Airplane<T>::input ()
{
    while (m_power)
       pop_front ();
    int power;
    std::cout << "Enter number of models:";
    std::cin >> power;
    int counter = 1;
    T curr;
    while( counter <= power ) {
```

```
        std::cout << "Enter element";
        std::cin >> curr;
        pushBack(curr);

         counter++;
    }
}

template<typename T>
void Airplane<T>::pop_front ()
{
    Node<T> *temp = m_model;
    m_model = m_model->pNext;
    delete temp;
    m_power--;
}

template<typename T>
void Airplane<T>::pushBack (T element)
{
    if( m_model== nullptr ) {
        m_model = new Node<T>(element);

     }
    else {
        Node<T> *curr = this->m_model;
        while( curr->pNext != nullptr ) {
```

```
            curr = curr->pNext;
        }
        curr->pNext = new Node<T>(element);

    }
    m_power++;
}

template<typename T>
T Airplane<T>::getModel (int index)
{
    int counter = 0;
    Node<T> *curr = this->m_model;
    while( curr != nullptr ) {
        if( counter == index ) {
            return curr->info;
        }
        curr = curr->pNext;
        counter++;
    }

}

template<typename T>
Airplane<T>::~Airplane ()
{
    while( m_model != nullptr ) {
```

```
        pop_front();
    }
}



int main()
{
    char a[] = { 'b', 'b' };
    Airplane<char> titan(a,2);  // створюєм обєкт титан з параметрами масив а та розмір 2
    titan.pushBack ('k');// додаєм елемент к в кінець титану
    Airplane<char> brain(titan); // створюєм обєкт brain ідентичний титану
    brain.setModel (a,2); //додаєм масив в кінець обєкта
    std::cout << "Titan is ";
    titan.print ();  // виводим титан на консоль
    std::cout << "Second element of titan:" << titan.getModel(1) << std::endl;
    brain.pushBack ('@');
    std::cout << "Brain is";
     brain.print();
     brain.input (); // очищуєм обєкт та вводим нові значення вручну
     std::cout << "Brain is"; // виводим обєкт
     brain.print ();
}
```

Результат виконання програми:



Висновок: Я навчилася працювати з лінійними списками. Будувати конструктор і деструктор класу