Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа 5

"Перегрузка операций"

Выполнил:
студент 2 курса
группы АС-53
Зайчук Д.Р.
Проверила:
Давидюк Ю.И.

Брест, 2020

Вариант 10

1) Постановка задачи. АТД - однонаправленный список с элементами типа char.
   Дополнительно перегрузить следующие операции:
   - '[]' - доступ к элементу в заданной позиции;
   - '+' - объединить два списка;
   - '==' - проверка на равенство.
2) Определение класса.
   List.h

```
#pragma once
#include <iostream>
#include <cstring>

using namespace std;

struct Element {
    char data;
    Element* next;
};

const int MAX = 20;

class List {
    Element* pHead;
    Element* pPrev;
    int size = 0;

public:
    List();
    ~List();
    void addToList(char data);
    void printList();
    Element* operator [](int i);
    friend List operator +(List&, List&);
    friend bool operator ==(List&, List&);
};
```

3) Реализация класса.

List.cpp

```
#include "List.h"
List::List() {
      pHead = NULL;
      pPrev = NULL;
      cout << endl << "---Constructor---" << endl;
}
```

```
List::~List() {
    cout << endl << "---Destructor---" << endl;
}

void List::addToList(char data) {
      Element* temp = new Element;
      if (pHead == NULL) {
            pHead = temp;
            temp->data = data;
            temp->next = NULL;
            pPrev = temp;
            size++;
      }
      else if (size < MAX) {
            pPrev->next = temp;
            temp->data = data;
            temp->next = NULL;
            pPrev = temp;
            size++;
      }
      else {
            cout << "Max size reached (20)" << endl;
      }
}

void List::printList() {
      Element* pTemp = pHead;
      while (pTemp != NULL) {
            std::cout << pTemp->data << " ";
            pTemp = pTemp->next;
      }
}

Element* List::operator [](int i) {
      if(i < 0 && i > MAX) {
            cout << "Invalid index" << endl;
      }
      else {
            Element* key1 = this->pHead;
            Element* key2;
            for (int j = 0; j < i; j++) {
                  key2 = key1->next;
                  key1 = key2;
            }
            return key1;
      }
}

List operator +(List& l1, List& l2) {
      Element* temp = new Element;
      int h = l1.size;
      for (int i = h; i < MAX; i++) {
            if (l2.pHead != NULL) {
                  char a = l2.pHead->data;
                  l1.addToList(a);
```

```
                Element* pTemp = l2.pHead;
                pTemp = pTemp->next;
                l2.pHead = pTemp;
                l2.size--;
            }
            else {
                return l1;
            }
        }
        cout << "Max size reached (20)" << endl;
        return l1;
    }

    bool operator ==(List& l6, List& l7) {
        Element* p1Temp = l6.pHead;
        Element* p2Temp = l7.pHead;
        while (p1Temp != NULL) {
            if (p1Temp->data == p2Temp->data){
                p1Temp = p1Temp->next;
                p2Temp = p2Temp->next;
            }
            else {
                return false;
            }
        }
        return true;
    }
```

4) Листинг основной программы.

main.cpp

```
#include <iostream>
#include <cstring>
#include "List.h"

using namespace std;

int main(void) {
    char x;
    List a, b;

    cout << "Enter element of the first list " << endl;
    int i = 0; bool add = true;
    while (add) {
        i++;
        cout << "Enter element ";
        cin >> x;
        a.addToList(x);
        if (i < MAX) {
            cout << "Add one more?(y/n) ";
            cin >> x;
            if (x != 'y')
                add = false;
        }
```

```
		else {
			cout << "Max size reached" << endl;
			add = false;
		}
	}

	cout << "Enter element of the secend list " << endl;
	i = 0; add = true;
	while (add) {
		i++;
		cout << "Enter element ";
		cin >> x;
		b.addToList(x);
		if (i < MAX) {
			cout << "Add one more?(y/n) ";
			cin >> x;
			if (x != 'y')
				add = false;
		}
		else {
			cout << "Max size reached" << endl;
			add = false;
		}
	}

	cout << endl << "First list " << endl;
	a.printList();
	cout << endl << "Second list " << endl;
	b.printList();

	cout << "Second element of first list: " << endl;
	cout << a[1]->data << endl;
	if (a == b) {
		cout << "Lists are equal" << endl;
	}
	else {
		cout << "Lists aren't equal" << endl;
	}
	cout << "Sum of lists: " << endl;
	(a + b).printList();
}
```

Выбор способа перегрузки операций обоснован желанием попробовать в практической реализации разных методов.

Представление элемента списка в виде структуры обусловлено необходимостью хранить в одном объекте и данные, и указатель на следующий элемент.

Результат выполнения программы:

```
---Constructor---

---Constructor---
Enter element of the first list
Enter element a
Add one more?(y/n) y
Enter element b
Add one more?(y/n) y
Enter element c
Add one more?(y/n) n
Enter element of the secend list
Enter element g
Add one more?(y/n) n

First list
a b c
Second list
g Second element of first list:
b
Lists aren't equal
Sum of lists:
a b c g
---Destructor---

---Destructor---

---Destructor---
```

Вывод:

В ходе выполнения работы я получил практические навыки создания абстрактных типов данных и перегрузки операций в языке C++.