Министерство образования Республики Беларусь

Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №3.
"Перегрузка операций"

Выполнил:
Ст. 2 курса гр. АС-53
Демидович А. Г.
Проверила:
Давидюк Ю. И.

Брест, 2020

(Вариант 8)

8. АТД – однонаправленный список с элементами типа **char.** Дополнительно перегрузить следующие операции:

+ – добавить элемент в начало(char+list);

-- – удалить элемент из начала(типа –list);

== – проверка на равенство.

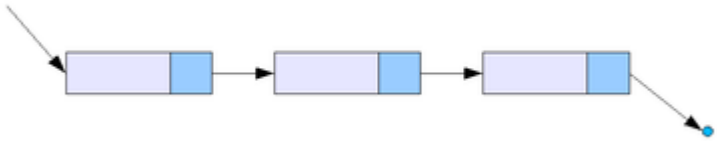
Определение класса:

```
#pragma once

// Ячейка
struct cell
{
	char value = 0;
	cell* next = nullptr;
};

class clist // Односвязный список
{
private:
	cell* begin;
	int size;
public:
	clist() : begin(nullptr), size(0) { }	// Конструктор
	clist(const clist&); // Копирования
	~clist(); // Деструктор
	void push(char _value); // Добавление элемента в список
	void pop(); // Удаление элемента списка
	void popAll(); // Удаление всех элементов списка
	void print();
	inline int getSize() { return size; } // Получение количества элементов, находящихся в
списке
	bool equal(const clist&);

	clist& operator+=(char); // Перегрузка оператора +=
	friend clist operator+(clist&, char); // Перегрузка оператора +
	clist operator--(); // Перегрузка оператора --
	bool operator==(const clist&);
};

#include "clist.h"
#include <iostream>


clist::clist(const clist& _list)
{
	cell* temp = _list.begin;
	for (int i = 0; i < _list.size; i++) {
		push(temp->value);
		temp = temp->next;
	}
	size = _list.size;
	begin = _list.begin;
	delete(temp);
}

clist::~clist() {
```

```
}

void clist::push(char _value) {
	cell* newCell = new cell;
	newCell->value = _value;
	newCell->next = NULL;

	if (begin == nullptr) {
		begin = newCell;
		size++;
		return;
	}
	newCell->next = begin;
	begin = newCell;
	size++;
}

void clist::pop() {
	cell* cur = begin;
	begin = begin->next;
	size--;
	delete cur;
	cur = nullptr;
}

void clist::popAll()
{
	while (size != 0) // Пока еще есть элементы
		pop(); // Удаляем элементы по одному
}

void clist::print() {
	cell* temp = begin;
	while (temp != nullptr) {
		std::cout << temp->value << "   ";
		if (temp->next == NULL)
			break;
		temp = temp->next;
	}
	std::cout << std::endl;
}

bool clist::equal(const clist& _clist) {
	if (size != _clist.size)
		return 0;
	cell* tf = this->begin;
	cell* tf2 = _clist.begin;
	for (int i = 0; i < size; i++) {
		if (tf->value != tf2->value)
			return 0;
		tf = tf->next;
		tf2 = tf2->next;
	}
	return 1;
}
```

Объяснить выбранное представление памяти для объектов реализуемого класса.

Данные представлены в виде односвязного списка.



Реализация перегруженных операций с обоснованием выбранного способа (функция – член класса, внешняя функция, внешняя дружественная функция).

```
clist& clist::operator+=(char _value) {
	push(_value);
	return *this;
}

clist clist::operator--() {
	pop();
	return *this;
}

bool clist::operator==(const clist& _clist) {
	return equal(_clist);
}
```

Перегруженные операторы являются членами класса, т.к. должны вызываться объектом класса.

```
clist operator+(clist& _list, char _value)
{
	clist _bufferlist = _list;
	_bufferlist += _value;
	return _bufferlist;
}
```

Данный оператор – дружественный, не является членом класса, не вызывается объектом.

Тестовая программа:

```
#include <iostream>
#include "clist.h"

int main()
{
	clist list1, list2;
	list1 += 'A';
	list1 = list1 + 'D';

	list2 += 'A';
	list2 = list2 + 'C';
	list2 = list2 + 'D';

	list1.print();
	list2.print();
	std::cout << "\n Delete elements. \n" << std::endl;

	--list1;
	--list2;
	--list2;

	list1.print();
	list2.print();
```

```
	if (list1 == list2)
		std::cout << "Lists equal" << std::endl;
	else
		std::cout << "Lists not equal" << std::endl;

	return 0;
}
```

```
D   A
D   C   A

 Delete elements.

A
A
Lists equal
```