Министерство образования Республики Беларусь

Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

# Лабораторная работа №5.
# "Перегрузка операций"

Выполнил:
Ст. 2 курса гр. АС-53
Бранчук Д. В
Проверила:
Давидюк Ю. И.

Брест, 2020

**1. Цель.** Получить практические навыки создания абстрактных типов данных и перегрузки операций в языке C++.

## 2. Постановка задачи (Вариант 3)

3. АТД – множество с элементами типа **char.** Дополнительно перегрузить следующие операции:

- – удалить элемент из множества (типа set-char);

> – проверка на подмножество;

!= – проверка множеств на неравенство.

## 3. Определение класса:

```
#ifndef MYSET_H
#define MYSET_H

class myset {
private:
	char* value;
	int count;
public:
	myset() : count(0), value(nullptr) { }
	myset(const myset&);
	~myset();
	inline bool empty() const { return count == 0; }
	inline char getChar(int position) const { return value[position]; }
	inline int size() const { return count; }
	void push(const char item);
	void remove(const char item);
	void print();
	void input(int size);
	bool isEqual(const myset& mset);
	bool subset(const char item);
	bool subset(const myset& mset);

	myset& operator-=(const char);
	myset& operator=(const myset&);
	bool operator!=(const myset&);
	bool operator==(const myset&);
	bool operator>(const myset&);
	friend myset operator-(const myset&, const char);
};

#endif // !MYSET
```

## Описание методов и функций класса:

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

myset::myset(const myset& mset) {
	try {
		value = new char[mset.count];
		count = mset.count;
		for (int i = 0; i < count; i++)
			value[i] = mset.value[i];
	}
```

```
	catch (std::bad_alloc e)
	{
		std::cout << e.what() << std::endl;
	}
}

void myset::input(int size) {
	char key;
	for (int k = 0; k < size; k++) {
		std::cout << "Enter element #" << k << ": ";
		std::cin >> key;
		this->push(key);
	}
}

void myset::print() {
	for (int i = 0; i < count; i++)
		std::cout << value[i] << "\t";
	std::cout << std::endl;
	std::cout << "Print Done.\n" << std::endl;
}

bool myset::subset(const char item) {
	for (int i = 0; i < count; i++) {
		if (value[i] == item)
			return 1;
	}
	return 0;
}

bool myset::subset(const myset& mset) {
	bool find = false;
	if (count >= mset.count) {
		for (int i = 0; i < mset.count; i++) {
			for (int k = 0; k < count; k++) {
				if (value[k] == mset.getChar(i)) {
					find = true;
				}
			}
			if (!find)
				return 0;
			find = false;
		}
		return 1;
	}
	else
		return 0;
}

bool myset::isEqual(const myset& mset) {
	if (count != mset.count)
		return 0;
	for (int i = 0; i < count; i++) {
		if (value[i] != mset.value[i])
			return 0;
	}
	return 1;
}

void myset::push(const char item)
{
```

```
	char* p2;
	p2 = value;
	bool isFind = false;

	try {
		if (subset(item))
			return;
		value = new char[count + 1];
		for (int i = 0; i < count; i++)
			value[i] = p2[i];
		for (int i = 0; i < count; i++) {
			if (item < value[i])
			{
				for (int k = count; k > i; k--)
				{
					value[k] = value[k - 1];
				}
				value[i] = item;
				isFind = true;
				break;
			}
		}
		if (!isFind)
			value[count] = item;
		count++;
		if (count > 0)
			delete[] p2;
	}
	catch (std::bad_alloc e) {
		std::cout << e.what() << std::endl;
		value = p2;
	}
}

myset::~myset() {
	if (count > 0)
		delete[] value;
}

void myset::remove(const char item) {
	if (count < 1)
		return;
	if (!subset(item))
		return;
	try {
		char* val2;
		val2 = new char[count - 1];
		for (int i = 0; i < count - 1; i++)
			if (value[i] != item)
				val2[i] = value[i];
			else
			{
				for (int k = i ; k < count - 1; k++)
					val2[k] = value[k + 1];
				break;
			}
		count--;
		if (count > 0)
			delete[] value;
		value = val2;
	}
```

```
	catch (std::bad_alloc e)
	{
		std::cout << e.what() << std::endl;
	}
}

myset& myset::operator=(const myset& obj) {
	char* val2;

	try {
		val2 = new char[obj.count];
		if (count > 0)
			delete[] value;
		value = val2;
		count = obj.count;
		for (int i = 0; i < count; i++)
			value[i] = obj.value[i];
	}
	catch (std::bad_alloc e)
	{
		std::cout << e.what() << std::endl;
	}
	return *this;
}

myset& myset::operator-=(const char item) {
	remove(item);
	return *this;
}

myset operator-(const myset& mset, const char item) {
	myset buff(mset);
	buff -= item;
	return buff;
}

bool myset::operator>(const myset& mset) {
	return subset(mset);
}

bool myset::operator!=(const myset& mset) {
	return !isEqual(mset);
}

bool myset::operator==(const myset& mset) {
	return !isEqual(mset);
}
```

## 4. Обоснование включения в класс нескольких конструкторов, деструктора и операции присваивания:

- `myset::myset(const myset& mset)` – **конструктор копирования, требуется для корректного создания объекта, на основе уже существующего.**
- `myset() : count(0), value(nullptr)` - **конструктор, создающий объект без заданных параметров.**
- **`~myset()` – деструктор, очищает массив `char*`**

- `myset& myset::operator=` – **Оператор присваивания, требуется для корректного создания копии.**

## 5. Объяснить выбранное представление памяти для объектов реализуемого класса.

Значения множества хранятся в динамическом массиве char* это требуется для корректного добавления и удаления элементов в множество.

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

- ```
  myset& myset::operator-=(const char item) {
        remove(item);
        return *this;
  }
  myset operator-(const myset& mset, const char item) {
        myset buff(mset);
        buff -= item;
        return buff;
  }
  ```

Данные перегруженные операторы - члены класса, что мы изменяем приватные поля класса и используем приватные поля другого объекта класса, поэтому необходимо, чтобы они являлись членами класса.

```
bool myset::operator>(const myset& mset) {
      return subset(mset);
}

bool myset::operator!=(const myset& mset) {
      return !isEqual(mset);
}

bool myset::operator==(const myset& mset) {
      return !isEqual(mset);
}
```

Данные перегруженные операторы являются дружественными, мы не изменяем приватные поля передаваемых объектов класса, а создаем и возвращаем новый объект, именно для этого и используется ключевое слово friend.

## 6. Тестовая программа:

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

int main()
{
      myset set1;
      myset set2;
      myset set3;
      set1.input(2);
      set2.input(2);
      set3.input(2);
      set1.print();
      set2.print();
```
```

```
	set1 = set1 - 'A';
	set2 = set2 - 'B';
	set1.print();
	set2.print();
	if (set1 != set2)
		std::cout << "Sets not equal." << std::endl;
	else
		std::cout << "Sets equal." << std::endl;
	if (set3 > set2)
		std::cout << "Set2 is subset of set3" << std::endl;
	else
		std::cout << "Set2 isn't subset of set3" << std::endl;
	return 0;
}
```

```
Enter element #0: A
Enter element #1: B
Enter element #0: A
Enter element #1: B
Enter element #0: A
Enter element #1: B
A       B
Print Done.

A       B
Print Done.

B
Print Done.

A
Print Done.

Sets not equal.
Set2 is subset of set3
```

6. Вывод:

Получил практические навыки создания абстрактных типов данных и перегрузки операций в языке C++.