Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

# Лабораторная работа №3. "Перегрузка операций"

Выполнил:

Ст. 2 курса гр. АС-53

Альциванович Н. В

Проверила:

Давидюк Ю. И.

Брест, 2020

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

**2. Постановка задачи (Вариант 1)**

1. АТД − множество с элементами типа **char.** Дополнительно перегрузить следующие операции:

+ − добавить элемент в множество(типа char + set);

+ − объединение множеств;

= = − проверка множеств на равенство.

**3. Определение класса:**

#ifndef CSET\_H

#define CSET\_H

class cset {

private:

char\* elements;

int count;

public:

cset() : count(0), elements(nullptr) { }

cset(const cset&);

~cset();

inline bool empty() const { return count == 0; }

inline char getChar(int position) const { return elements[position]; }

inline int size() const { return count; }

void push(const char);

void print(const char\*);

void input(int size);

bool subchar(const char);

bool isEqual(const cset&);

cset& operator+=(const char);

cset& operator+=(const cset&);

cset& operator=(const cset&);

bool operator==(const cset&);

friend cset operator+(const char, const cset&);

friend cset operator+(const cset&, const cset&);

};

#endif // !CSET

**Описание методов и функций класса:**

#include "cset.h"

#include <iostream>

cset::cset(const cset& mset) {

try {

elements = new char[mset.count];

count = mset.count;

for (int i = 0; i < count; i++)

elements[i] = mset.elements[i];

}

catch (std::bad\_alloc e)

{

std::cout << e.what() << std::endl;

}

}

void cset::input(int size) {

char key;

for (int k = 0; k < size; k++) {

std::cout << "Enter char #" << k << ": ";

std::cin >> key;

this->push(key);

}

std::cout << "\n" << std::endl;

}

void cset::print(const char\* name) {

std::cout << name << ": " << std::endl;

for (int i = 0; i < count; i++)

std::cout << elements[i] << "\t";

std::cout << "\n" << std::endl;

}

bool cset::subchar(const char item) {

for (int i = 0; i < count; i++) {

if (elements[i] == item)

return 1;

}

return 0;

}

bool cset::isEqual(const cset& \_cset) {

if (count != \_cset.count)

return 0;

for (int i = 0; i < count; i++) {

if (elements[i] != \_cset.elements[i])

return 0;

}

return 1;

}

void cset::push(const char \_element)

{

char\* p2;

p2 = elements;

bool isFind = false;

try {

if (subchar(\_element))

return;

elements = new char[count + 1];

for (int i = 0; i < count; i++)

elements[i] = p2[i];

for (int i = 0; i < count; i++) {

if (\_element < elements[i])

{

for (int k = count; k > i; k--)

{

elements[k] = elements[k - 1];

}

elements[i] = \_element;

isFind = true;

break;

}

}

if (!isFind)

elements[count] = \_element;

count++;

if (count > 0)

delete[] p2;

}

catch (std::bad\_alloc e) {

std::cout << e.what() << std::endl;

elements = p2;

}

}

cset::~cset() {

if (count > 0)

delete[] elements;

}

cset& cset::operator=(const cset& obj) {

char\* val2;

try {

val2 = new char[obj.count];

if (count > 0)

delete[] elements;

elements = val2;

count = obj.count;

for (int i = 0; i < count; i++)

elements[i] = obj.elements[i];

}

catch (std::bad\_alloc e)

{

std::cout << e.what() << std::endl;

}

return \*this;

}

cset& cset::operator+=(const char \_element) {

push(\_element);

return \*this;

}

cset& cset::operator+=(const cset& \_cset) {

for (int i = 0; i < \_cset.count; i++)

push(\_cset.elements[i]);

return \*this;

}

cset operator+(const char \_element, const cset& \_cset) {

cset csethelp(\_cset);

csethelp += \_element;

return csethelp;

}

cset operator+(const cset& \_cset, const cset& \_cset2) {

cset csethelp(\_cset);

csethelp += \_cset2;

return csethelp;

}

bool cset::operator==(const cset& \_cset) {

return !isEqual(\_cset);

}

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

* cset::cset(const cset& \_cset) **– конструктор копирования, требуется для корректного создания объекта, на основе уже существующего.**
* cset() : count(0), elements(nullptr) – **конструктор, создающий объект без заданных параметров.**
* ~cset() **– деструктор, очищает массив char\***
* cset& cset::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).**

* cset& cset::operator+=(const char \_element) {

push(\_element);

return \*this;

}

cset& cset::operator+=(const cset& \_cset) {

for (int i = 0; i < \_cset.count; i++)

push(\_cset.elements[i]);

return \*this;

}

Данные перегруженные операторы - члены класса, т.к. изменяем приватные поля класса и используем приватные поля другого объекта класса.

cset operator+(const char \_element, const cset& \_cset) {

cset csethelp(\_cset);

csethelp += \_element;

return csethelp;

}

cset operator+(const cset& \_cset, const cset& \_cset2) {

cset csethelp(\_cset);

csethelp += \_cset2;

return csethelp;

}

bool cset::operator==(const cset& mset) {

return isEqual(mset);

}

Данные перегруженные операторы являются дружественными, для доступа к приватным полям обоих объектов класса.

**6. Тестовая программа:**

#include <iostream>

#include "cset.h"

int main()

{

cset set1;

cset set2;

cset set3;

set1.input(2);

set2.input(2);

set1.print("set1");

set2.print("set2");

set1 = 'Y' + set1;

set2 = 'Z' + set2;

set1.print("set1 + Y");

set2.print("set2 + Z");

set3 = set1 + set2;

set3.print("set1 + set2");

if (set1 == set2)

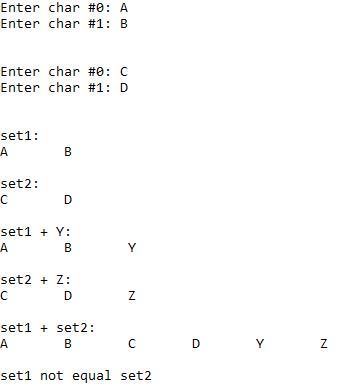
std::cout << "set1 equal set2" << std::endl;

else

std::cout << "set1 not equal set2" << std::endl;

return 0;

}



6. Вывод:

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