Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Уфимский университет науки и технологий»

Факультет информатики и робототехники

Кафедра Информатики

Отчёт по лабораторной работе №3

По дисциплине «Объектно-ориентированное программирование»

Выполнил:

Студент группы ПРО-231 К.И. Арменшин

Проверил:

доцент кафедры ВМиК Котельников В.А.

Уфа – 2023

**Цель:**

Определение и реализация класса-контейнера разнообразных объектов (принадлежащих различным классам, имеющим общего предка), и написание программы, иллюстрирующей использование контейнера.

**Ход работы:**

Функции контейнера объектов

* добавление объектов (в начало, в конец), вставка объектов (в середину)
* void pushBack(Type value); // Помещенеи элемнта в конец контейнера
* void pushFront(Type value); // Помещенеи элемнта в начало контейнера
* void insertValue(Type value, int index); // Помещенеи элемнта в позицию с индексом "index" в контейнер
* изъятие объектов (с удалением самого объекта и без)
* Type topFront(); // Получение элемента из начало контейнера
* Type popFront(); // Получение элемента из начало контейнера с последующим удалением
* Type topBack(); // Получение элемента из конца контейнера
* Type popBack(); // Получение элемента из конца контейнера с последующим удалением
* переход по объектам, если это применимо (текущий, предыдущий, последующий, проверка наличия)
* void getNext() { // Следующий элемент
* void getPrev() { // Предыдущий элемент
* bool notEnd() { // Проверка на конец
* получение очередного объекта из контейнера или объекта по индексу
* Type getValue(int index) { // Получение значения элемента
* поочередное обращение к каждому объекту контейнера; вызов функций, реализуемых всеми классами объектов контейнера

Функции основной программы: код, который в случайном порядке:

* создаёт объекты
* добавляет в контейнер
* использует (для упрощённого вывода)
* удаляет из контейнера

int rd = rand() % 8;

switch (rd) {

case 0: // Добавление в начало

store.pushFront(rand());

break;

case 1: // Добавление в конец

store.pushBack(rand());

break;

case 2: // Вставка

if (store.GetSize())

store.insertValue(rand(), rand() % store.GetSize());

else

store.insertValue(rand(), 0);

break;

case 3: // Получение первого элемента

store.topFront();

break;

case 4: // Получение последнего элемента

store.topBack();

break;

case 5: // Удаление из начала

if (store.GetSize())

store.popFront();

break;

case 6: // Удаление из конца

if (store.GetSize())

store.popBack();

break;

case 7: // Удаление по индексу

if (store.GetSize())

store.erase(rand() % store.GetSize());

break;

}

Вывод результатов тестирование на 100, 1000 и 100000 случайных действий.

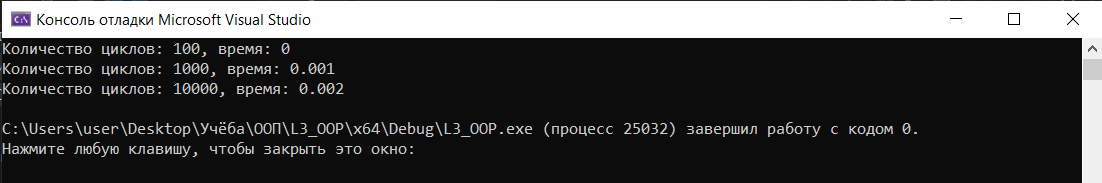
****

Рисунок 1. Пример работы программы

**Вывод**

В результате выполнения данной лабораторной работы была разработана программа, реализующая контейнер с возможностью добавления элементов, удаления элементов, изъятия элементов, перехода по элементам, получения значения элемента.

Полный код находится в Github:

https://github.com/karimarm/Labs\_OOP.git

**Приложение 1**

L3\_ECG.cpp

#include <iostream>

#include <cassert>

using namespace std;

template<class Type>

struct Cell

{

Type value;

Cell<Type>\* left = nullptr;

Cell<Type>\* right = nullptr;

};

template<class Type>

ostream& operator<< (ostream& out, const Cell<Type>& cell) {

return (out << cell.value);

}

template<class Type>

class Storage {

private:

int size; // Размер контейнера

Cell<Type>\* firstCell; // Указатель на начальный элемент

Cell<Type>\* lastCell; // Указатель на последний элемент

public:

Storage() { // Конструктор

//cout << "Storage() " << this << endl;

size = 0;

firstCell = lastCell = nullptr;

}

int GetSize() { // Получение размера контеёнера

return size;

}

Cell<Type>\* first() { // Получение первого элемента

return firstCell;

}

Cell<Type>\* last() { // Получение последнего элемента

return lastCell;

}

Type getValue(int index) { // Получение значения элемента

return position(index)->value;

}

Cell<Type>\* position(int index); // Получение указателя на элемент контейнера с индексом "index"

void pushBack(Type value); // Помещенеи элемнта в конец контейнера

void pushFront(Type value); // Помещенеи элемнта в начало контейнера

void insertValue(Type value, int index); // Помещенеи элемнта в позицию с индексом "index" в контейнер

Type topFront(); // Получение элемента из начало контейнера

Type popFront(); // Получение элемента из начало контейнера с последующим удалением

Type topBack(); // Получение элемента из конца контейнера

Type popBack(); // Получение элемента из конца контейнера с последующим удалением

Type erase(int index); // Удаление элемента с индексом "индекс" из контейнера

Type erase(Cell<Type>\* cell); // Удаление элемента "cell" из контейнера

void printStorage() {

for (Cell<Type>\* it = firstCell; it != nullptr; it = it->right)

{

cout << it->value << " - " << it << " - " << it->left << " - " << it->right << endl;

}

}

~Storage() { // Деструктор

if (!size)

return;

Cell<Type>\* iter = this->firstCell;

for (Cell<Type>\* it = iter->right; it != nullptr; it = it->right)

{

delete iter;

iter = it;

}

if (iter != nullptr)

delete iter;

}

};

template<class Type>

class StorageIterator {

private:

Storage<Type>\* store;

Cell<Type>\* iter;

public:

StorageIterator(Storage<Type>& store) : store(&store), iter(nullptr) {} // Конструктор

void first() { // Первый элемент

iter = store->first();

}

void last() { // Последний элемент

iter = store->last();

}

void getNext() { // Следующий элемент

iter = iter->right;

}

void getPrev() { // Предыдущий элемент

iter = iter->left;

}

void getPosition(int index) { // Элемент на позиции "index"

iter = store->position(index);

}

bool notEnd() { // Проверка на конец

return (iter != nullptr);

}

Type getValue() { // Получение значения

return iter->value;

}

};

void test(int quantcity) {

srand(time(NULL));

clock\_t time\_start = clock();

Storage<int> store;

int coun = quantcity;

while (coun--) {

int rd = rand() % 8;

switch (rd) {

case 0: // Добавление в начало

store.pushFront(rand());

break;

case 1: // Добавление в конец

store.pushBack(rand());

break;

case 2: // Вставка

if (store.GetSize())

store.insertValue(rand(), rand() % store.GetSize());

else

store.insertValue(rand(), 0);

break;

case 3: // Получение первого элемента

store.topFront();

break;

case 4: // Получение последнего элемента

store.topBack();

break;

case 5: // Удаление из начала

if (store.GetSize())

store.popFront();

break;

case 6: // Удаление из конца

if (store.GetSize())

store.popBack();

break;

case 7: // Удаление по индексу

if (store.GetSize())

store.erase(rand() % store.GetSize());

break;

}

}

clock\_t time\_finish = clock();

double time\_delta = (double)(time\_finish - time\_start) / CLOCKS\_PER\_SEC;

cout << "Количество циклов: " << quantcity << ", время: " << time\_delta << endl;

}

int main()

{

setlocale(LC\_ALL, "ru");

test(100);

test(1000);

test(10000);

return 0;

}

template<class Type>

Cell<Type>\* Storage<Type>::position(int index) { // Получение указателя на элемент контейнера с индексом "index"

Cell<Type>\* iter = firstCell;

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

if (i == index) {

return iter;

}

else {

iter = iter->right;

}

}

return nullptr;

}

template<class Type>

void Storage<Type>::pushBack(Type value) { // Помещенеи элемнта в конец контейнера

Cell<Type>\* cell = new Cell<Type>;

cell->value = value;

cell->left = lastCell;

cell->right = nullptr;

if (size) lastCell->right = cell;

lastCell = cell;

if (!size) firstCell = cell;

size++;

}

template<class Type>

void Storage<Type>::pushFront(Type value) { // Помещенеи элемнта в начало контейнера

Cell<Type>\* cell = new Cell<Type>;

cell->value = value;

cell->left = nullptr;

cell->right = firstCell;

if (size) firstCell->left = cell;

firstCell = cell;

if (!size) lastCell = cell;

size++;

}

template<class Type>

void Storage<Type>::insertValue(Type value, int index) { // Помещенеи элемнта в позицию с индексом "index" в контейнер

if (index == 0) {

pushFront(value);

return;

}

if (index == size) {

pushBack(value);

return;

}

if (index < 0 || index > size) {

cout << "Неверный индекс!\n";

return;

}

Cell<Type>\* it = position(index - 1);

Cell<Type>\* cell = new Cell<Type>;

it->right->left = cell;

cell->value = value;

cell->left = it;

cell->right = it->right;

it->right = cell;

size++;

}

template<class Type>

Type Storage<Type>::topFront() { // Получение элемента из начало контейнера

if (size)

return firstCell->value;

}

template<class Type>

Type Storage<Type>::popFront() { // Получение элемента из начало контейнера с последующим удалением

if (size < 1) {

cout << "Невозможно удалить элемент, так как контейнер пуст!\n";

return 0;

}

Type val = firstCell->value;

if (size == 1) {

delete firstCell;

firstCell = lastCell = nullptr;

size--;

return val;

}

firstCell = firstCell->right;

delete firstCell->left;

firstCell->left = nullptr;

size--;

return val;

}

template<class Type>

Type Storage<Type>::topBack() { // Получение элемента из конца контейнера

if (size)

return lastCell->value;

}

template<class Type>

Type Storage<Type>::popBack() { // Получение элемента из конца контейнера с последующим удалением

if (size < 1) {

cout << "Невозможно удалить элемент, так как контейнер пуст!\n";

return 0;

}

Type val = lastCell->value;

if (size == 1) {

delete lastCell;

firstCell = lastCell = nullptr;

size--;

return val;

}

lastCell = lastCell->left;

delete lastCell->right;

lastCell->right = nullptr;

size--;

return val;

}

template<class Type>

Type Storage<Type>::erase(int index) { // Удаление элемента с индексом "индекс" из контейнера

assert(index >= 0 && index <= size);

if (index == 0) {

return popFront();

}

if (index == size - 1) {

return popBack();

}

Cell<Type>\* cell = position(index);

Type val = cell->value;

(cell->left)->right = cell->right;

(cell->right)->left = cell->left;

delete cell;

size--;

return val;

}

template<class Type>

Type Storage<Type>::erase(Cell<Type>\* cell) { // Удаление элемента "cell" из контейнера

if (cell == firstCell) {

return popFront();

}

if (cell == lastCell) {

return popBack();

}

Type val = cell->value;

(cell->left)->right = cell->right;

(cell->right)->left = cell->left;

delete cell;

size--;

return val;

}