**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 6**

Тема: Основы работы с коллекциями: аллокаторы

Студент: Калугин Кирилл Алексеевич

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи

Создать шаблон динамической коллекцию, согласно варианту задания:

1. Коллекция должна быть реализована с помощью умных указателей (std::shared\_ptr, std::weak\_ptr). Опционально использование std::unique\_ptr.

2. В качестве параметра шаблона коллекция должна принимать тип данных.

3. Коллекция должна содержать метод доступа:

o Стек – pop, push, top;

o Очередь – pop, push, top;

o Список, Динамический массив – доступ к элементу по оператору [].

4. Реализовать аллокатор, который выделяет фиксированный размер памяти (количество блоков памяти – является параметром шаблона аллокатора). Внутри аллокатор должен хранить указатель на используемый блок памяти и динамическую коллекцию указателей на свободные блоки. Динамическая коллекция должна соответствовать варианту задания (Динамический массив, Список, Стек, Очередь).

5. Коллекция должна использовать аллокатор для выделения и освобождения памяти для своих элементов.

6. Аллокатор должен быть совместим с контейнерами std::map и std::list (опционально – vector).

7. Реализовать программу, которая:

o позволяет вводить с клавиатуры фигуры (с типом int в качестве параметра шаблона фигуры) и добавлять в коллекцию использующую аллокатор;

o позволяет удалять элемент из коллекции по номеру элемента;

o выводит на экран введенные фигуры c помощью std::for\_each.

Вариант - пятиугольник, стэк, линейный список.

1. Описание программы

Программа реализует линейный аллокатор при помощи линейного списка и массива. В списке хранятся ссылки на блоки памяти, находящиеся в массиве. Класс стэка использует команду new для выделения памяти по ссылке. Таким образом динамическая память не используется. Взаимодействие с пользователем осуществляется посредством интерактивного меню. Также, программа реализует нетипичную для стэка функцию удаления элемента по номеру.

1. Набор тестов

PS C:\Users\Кирилл\Desktop\ООП\_6> g++ 1.cpp

PS C:\Users\Кирилл\Desktop\ООП\_6> .\a.exe

1.Execute command "push"

2.Execute command "pop"

3.Execute command "top"

4.Show stack

5.Delete element by number

0.Exit

Choose option: 1

Enter center coordinates and radius: 1 1 1

Choose option: 4

----------

A = (1,2)

B = (1.88295,1.46947)

C = (1.71934,0.305342)

D = (0.561629,0.101206)

E = (0.00973193,1.13917)

----------

Choose option: 1

Enter center coordinates and radius: 2 2 2

Choose option: 3

----------

A = (2,4)

B = (3.7659,2.93894)

C = (3.43868,0.610683)

D = (1.12326,0.202412)

E = (0.0194639,2.27835)

----------

Choose option: 2

Choose option: 3

----------

A = (1,2)

B = (1.88295,1.46947)

C = (1.71934,0.305342)

D = (0.561629,0.101206)

E = (0.00973193,1.13917)

----------

Choose option: 1

Enter center coordinates and radius: 3 3 3

Choose option: 1

Enter center coordinates and radius: 4 4 4

Choose option: 4

----------

A = (4,8)

B = (7.53179,5.87789)

C = (6.87736,1.22137)

D = (2.24652,0.404824)

E = (0.0389277,4.55669)

----------

----------

A = (3,6)

B = (5.64884,4.40841)

C = (5.15802,0.916025)

D = (1.68489,0.303618)

E = (0.0291958,3.41752)

----------

----------

A = (1,2)

B = (1.88295,1.46947)

C = (1.71934,0.305342)

D = (0.561629,0.101206)

E = (0.00973193,1.13917)

----------

Choose option: 5

Enter number: 2

Choose option: 4

----------

A = (3,6)

B = (5.64884,4.40841)

C = (5.15802,0.916025)

D = (1.68489,0.303618)

----------

----------

A = (1,2)

B = (1.88295,1.46947)

C = (1.71934,0.305342)

D = (0.561629,0.101206)

E = (0.00973193,1.13917)

----------

Choose option: 0

1. Листинг программы

//Калугин Кирилл Алексеевич М8О-207Б-19

//Создать шаблон динамической коллекцию, согласно варианту задания:

//1. Коллекция должна быть реализована с помощью умных указателей (std::shared\_ptr, std::weak\_ptr). Опционально использование std::unique\_ptr.

//2. В качестве параметра шаблона коллекция должна принимать тип данных.

//3. Коллекция должна содержать метод доступа:

//o Стек – pop, push, top;

//o Очередь – pop, push, top;

//o Список, Динамический массив – доступ к элементу по оператору [].

//4. Реализовать аллокатор, который выделяет фиксированный размер памяти (количество блоков памяти – является параметром шаблона аллокатора). Внутри аллокатор должен хранить указатель на используемый блок памяти и динамическую коллекцию указателей на свободные блоки. Динамическая коллекция должна соответствовать варианту задания (Динамический массив, Список, Стек, Очередь).

//5. Коллекция должна использовать аллокатор для выделения и освобождения памяти для своих элементов.

//6. Аллокатор должен быть совместим с контейнерами std::map и std::list (опционально – vector).

//7. Реализовать программу, которая:

// позволяет вводить с клавиатуры фигуры (с типом int в качестве параметра шаблона фигуры) и добавлять в коллекцию использующую аллокатор;

// позволяет удалять элемент из коллекции по номеру элемента;

// выводит на экран введенные фигуры с помощью std::for\_each.

//Вариант - пятиугольник; стэк; линейный список;

#include <iostream>

#include <memory>

#include <windows.h>

#include <algorithm>

#include <math.h>

#include <list>

#include <string>

const int inn = 100;

using namespace std;

const double PI = 3.141592653589793238462643383279502884197169399375105820974944;

template <class T, int number>

class Allocator {//класс аллокатора

public:

size\_t block\_size = sizeof (T);

size\_t total\_size;

int free = -1;

list <T\*> pointers;

T vault [number];

using value\_type = T;

using pointer = T \* ;

using const\_pointer = const T\*;

using size\_type = size\_t;

void init () {//функция инициализации аллокатора

total\_size = block\_size \* number;

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

pointers.push\_back (&vault [i]);

}

free = 0;

}

T\* allocate (int outn) {//функция выделения памяти в аллокаторе

T\* p;

if (free == -1) {

init ();

}

if ((free + 1) < number) {

p = pointers.front ();

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

pointers.pop\_front ();

free ++;

}

}

else {

throw "ERROR\nNo empty space in allocator!";

}

return p;

}

template <class U>

struct rebind {

using other = Allocator<U, number>;

};

};

template <typename T>

class Five {//класс пятиугольника

public:

using type = T;

pair<T, T> O;

T R;

Five () {};//конструктор класса

Five (T x, T y, T r): O(x, y), R(r) {};//конструктор класса

};

template <class T>//шаблон функции поиска вершин пятиугольника

void points (T E) {

Five <double> x = E.vault;

cout << "----------\n"<< "A = (" << (x.O).first << "," << (x.O).second + x.R << ")\n" << "B = ("

<< (x.O).first + x.R \* cos (28.0 / 180.0 \* PI) << "," << (x.O).second + x.R \* sin (28.0 / 180.0 \* PI)

<< ")\n" << "C = (" << (x.O).first + x.R \* cos (- 44.0 / 180.0 \* PI) << ","

<< (x.O).second + x.R \* sin (- 44.0 / 180.0 \* PI) << ")\n" << "D = ("

<< (x.O).first + x.R \* cos (- 116.0 / 180.0 \* PI) << "," << (x.O).second + x.R \* sin (- 116.0 / 180.0 \* PI)

<< ")\n" << "E = (" << (x.O).first + x.R \* cos (- 188.0 / 180.0 \* PI)

<< "," << (x.O).second + x.R \* sin (- 188.0 / 180.0 \* PI) << ")\n" << "----------\n";

}

template <class T>

class Elem {//класс элемента стэка

public:

T vault;

shared\_ptr <Elem <T>> prev;

};

template <class T>

class Iter {//класс итератора

public:

weak\_ptr <T> iter;

Iter <T>\* operator++ () {//перегруженный оператор ++

shared\_ptr <T> buffer = iter.lock();

if (buffer) {

iter = buffer->prev;

} else {

throw "Iterator out of range";

}

return (this);

}

T operator\* () {//перегруженный оператор разыменования

shared\_ptr <T> buffer = iter.lock();

return (\*(buffer.get ()));

}

bool operator!= (Iter <T> second) {//перегруженный оператор неравенства

shared\_ptr <T> b1 = iter.lock();

shared\_ptr <T> b2 = second.iter.lock();

bool ret;

if (b1.get() != b2.get()) {

ret = 1;

}

else {

ret = 0;

}

return (ret);

}

Iter () {};

Iter (shared\_ptr <T> elem) {

iter = elem;

};

};

template <class T, class Allocator>

class Stack {//класс стэка

public:

shared\_ptr <T> first;

Allocator alloc;

struct deleter{

void operator() (void\* ptr) {}

};

deleter del;

void push () {//функция вставки на первое место стэка

double x;

double y;

double r;

T\* b;

cout << "Enter center coordinates and radius: ";

cin >> x >> y >> r;

b = alloc.allocate (1);

shared\_ptr <T> curr (new (b) T, del);

curr->vault.O.first = x;

curr->vault.O.second = y;

curr->vault.R = r;

if (!first) {

first = curr;

} else {

curr->prev = first;

first = curr;

}

}

void pop () {//функция удаления первого элемента стэка

if (first) {

first = first->prev;

} else {

throw "ERROR\nStack is empty!";

}

}

void top () {//функция просмотра первого элемента

if (!first) {

cout << "Stack is empty!\n";

} else {

points (\*(first.get()));

}

}

Iter <T> begin () {//функция, возвращающая итератор на первый элемент

if (first) {

return (Iter <T> (first));

} else {

throw "Stack is empty!";

}

}

Iter <T> end () {//функция, возвращающая итератор на последний элемент

shared\_ptr <T> buffer = first;

while (buffer->prev) {

buffer = buffer->prev;

}

return (Iter <T> (buffer->prev));

}

void delBynumber (int numb) {//функция удаления по номеру

Iter <T> cur;

Iter <T> last;

cur = begin ();

last = end ();

if (numb = 1) {

pop ();

} else {

for (int i = 0; i < numb - 2; i++) {

if (cur != last) {

++ cur;

} else {

throw "ERROR\nNo such element in stack!";

}

}

cur.iter.lock()->prev = cur.iter.lock()->prev->prev;

}

}

};

int main () {

Stack <Elem <Five <double>>, Allocator <Elem <Five <double>>, inn>> S;

int option = 99;

cout << "1.Execute command \"push\"\n";

cout << "2.Execute command \"pop\"\n";

cout << "3.Execute command \"top\"\n";

cout << "4.Show stack\n";

cout << "5.Delete element by number\n";

cout << "0.Exit\n";

while (option != 0) {

cout << "Choose option: ";

cin >> option;

switch (option) {//меню

case 1: {

try {

S.push ();

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 2: {

try {

S.pop ();

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 3: {

try {

S.top ();

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 4: {

if (S.first) {

try {

if (S.begin () != S.end ()) {

if (S.first) {

for\_each (S.begin (), S.end (), points <Elem <Five <double>>>);

}

} else {

S.top ();

}

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

} else {

cout << "Stack is empty!\n";

}

break;

}

case 5: {

int x;

try {

cout << "Enter number: ";

cin >> x;

S.delBynumber (x);

}

catch (const char\* exc) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

cerr << exc << endl;

SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

}

break;

}

case 0: {

break;

}

default: {

cout << "Wrong command!\n";

break;

}

}

}

return(0);

}

1. Выводы

Я научился пользоваться аллокаторами и понял их внутреннее устройство. Также я узнал несколько новых типов аллокаторов.

Литература

# Аллокаторы памяти[Электронный ресурс]

URL:https://habr.com/ru/post/505632/ дата обращения (10.12.2020)

# Placement new, или как создать объект в выделенной памяти[Электронный ресурс]

URL:http://itw66.ru/blog/c\_plus\_plus/471.html дата обращения (10.12.2020)

# Использование аллокаторов[Электронный ресурс]

URL:https://ru.stackoverflow.com/questions/425224/Использование-аллокаторов дата обращения (10.12.2020)