минобрнауки россии

федеральное государственное бюджетное образовательное учреждение

высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |  |
| --- | --- |
| Институт (факультет) | Институт информационных технологий |
| Кафедра | Кафедра математического и программного обеспечения ЭВМ |

# РАЗРАБОТКА АБСТРАКТНЫХ ТИПОВ ДАННЫХ

|  |  |
| --- | --- |
| **Дисциплина:** | ООП |
| **Темы:** | Шаблоны |

**Среда разработки:** Microsoft Visual Studio

**Язык программирования:** C++

**Тип проекта:** Консольное приложение

**Лабораторная работа №4**

**Выполнил:** Овчинников М. В.

Череповец

2023

**ТРЕБОВАНИЯ К РАЗРАБОТКЕ**

1. Запрещается использовать обработку исключительных ситуаций и генерировать исключения.
2. Придерживайтесь принципа DRY (Don’t repeat yourself).
3. Обязательно наличие комментариев.
4. Обязательно сделать шаблонным класс-контейнер.

**ЦЕЛЬ РАБОТЫ**

Изучить способы определения шаблонов, принцип и механизм создания шаблонных классов и шаблонов функций, получить практические навыки работы с шаблонными классами.

**ЗАДАНИЯ**

*4 часа*

1. Ознакомьтесь с УМП по ООП часть 2 раздел VII.
2. Модифицируйте абстрактный типы данных, реализованные по заданию ЛР3, используя шаблоны определения класса и шаблоны определения функции.
3. Проверьте работоспособность АТД на тестовом наборе данных.

**ПРИМЕЧАНИЯ**

* Обязательно сделать шаблонным класс контейнер.
* Шаблоны определения функции на свое усмотрение.

Header.h

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma once

#include <iostream>

using namespace std;

enum class material { //Материал лодки

wood,

PWH,

plastic,

};

class boat //Класс для описании лодки

{

public:

boat();//Конструктор по умолчанию

boat(char\* boat\_name, material boat\_mat, double boat\_width, double boat\_length, double boat\_m, int boat\_load\_capacity, int boat\_power);//конструктор с параметрами

boat(material boat\_mat);//Конструктор с параметром material

boat(const boat& obj);//Конструктор копирующий

void print();

void set\_name(const char\* boat\_name);//Задает имя

char\* get\_name();//Возращает имя

void set\_mat(material boat\_mat);//задает материал

material get\_mat();//возращает материал

void set\_width(double boat\_width);//задает ширину

double get\_width();//возращает ширину

void set\_length(double boat\_length);//задает длину

double get\_length();//возращает длину

void set\_m( double boat\_m);//задает массу

double get\_m();//возращает массу

void set\_load\_capacity(int boat\_load\_capacity);//задает вместимость

int get\_load\_capacity();//возращает вместимость

void set\_power(int boat\_power);//задает мощность

int get\_power();//возращает мощность

static int get\_number\_of\_objects();//возращает кол-во объектов

friend std::ostream& operator<<(std::ostream& out, const boat& b);

// Перегрузка оператора равенства

bool operator==(const boat& b) const {

return (strcmp(this->name, b.name) == 0 &&

this->mat == b.mat &&

this->width == b.width &&

this->length == b.length &&

this->m == b.m &&

this->load\_capacity == b.load\_capacity &&

this->power == b.power);

}

// Перегрузка оператора неравенства

bool operator!=(const boat& b) const {

return !(\*this == b);

}

~boat();//деструктор

private:

char\* name;//имя

material mat;//материал

double width;//ширина

double length;//длина

double m;//масса

int load\_capacity;//вместимость (человек)

int power;//мощность (в лошадинных силах)

static int number\_of\_objects;//кол-во объектов

};

Source1.cpp

#include "Header.h"

int boat::number\_of\_objects = 0;

boat::boat():mat(material::PWH), width(0), length(0), m(0), load\_capacity(0), power(0){

name = NULL;

number\_of\_objects++;

}

boat::boat(char\* boat\_name, material boat\_mat, double boat\_width, double boat\_length, double boat\_m, int boat\_load\_capacity, int boat\_power):mat(boat\_mat), width(boat\_width), length(boat\_length), m(boat\_m), load\_capacity(boat\_load\_capacity), power(boat\_power)

{

name = new char[strlen(boat\_name) + 1];

strcpy(name, boat\_name);

number\_of\_objects++;

}

boat::boat(const boat& obj) :mat(obj.mat), width(obj.width), length(obj.length), m(obj.m), load\_capacity(obj.load\_capacity), power(obj.power) {

if (obj.name) {

name = new char[strlen(obj.name) + 1];

strcpy(name, obj.name);

}

else name = NULL;

number\_of\_objects++;

}

boat::boat(material boat\_mat):mat(boat\_mat),width(0), length(0), m(0), load\_capacity(0), power(0) {

name = NULL;

number\_of\_objects++;

}

void boat::set\_name(const char\* boat\_name) {

if (name) {

delete[] name;

name = nullptr;

}

if (boat\_name) {

name = new char[strlen(boat\_name) + 1];

strcpy(name, boat\_name);

}

}

char\* boat::get\_name() {

return name;

}

void boat::set\_mat(material boat\_mat) {

mat = boat\_mat;

}

material boat::get\_mat()

{

return mat;

}

void boat::set\_width(double boat\_width) {

boat::width = boat\_width;

}

double boat::get\_width() {

return width;

}

void boat::set\_length(double boat\_length) {

boat::length = boat\_length;

}

double boat::get\_length() {

return length;

}

void boat::set\_m(double boat\_m) {

boat::m = boat\_m;

}

double boat::get\_m()

{

return m;

}

void boat::set\_load\_capacity(int boat\_load\_capacity) {

boat::load\_capacity = boat\_load\_capacity;

}

int boat::get\_load\_capacity(){

return load\_capacity;

}

void boat::set\_power(int boat\_power) {

boat::power = boat\_power;

}

int boat::get\_power() {

return power;

}

int boat::get\_number\_of\_objects(){

return number\_of\_objects;

}

void boat::print() {

cout << this->get\_name() << endl;

if (this->get\_mat() == material::wood)

cout << "wood" << endl;

if (this->get\_mat() == material::PWH)

cout << "PWH" << endl;

if (this->get\_mat() == material::plastic)

cout << "plastic" << endl;

cout << this->get\_width() << endl;

cout << this->get\_length() << endl;

cout << this->get\_m() << endl;

cout << this->get\_load\_capacity() << endl;

cout << this->get\_power() << endl;

}

boat::~boat() {

delete[] name;

number\_of\_objects--;

}

std::ostream& operator<<(std::ostream& out, const boat& b) {

out << "Boat Name: " << (b.name)<< std::endl;

out << "Material: ";

switch (b.mat) {

case material::wood:

out << "Wood";

break;

case material::PWH:

out << "PWH";

break;

case material::plastic:

out << "Plastic";

break;

}

out << std::endl;

out << "Width: " << b.width << " meters" << std::endl;

out << "Length: " << b.length << " meters" << std::endl;

out << "Mass: " << b.m << " kg" << std::endl;

out << "Load Capacity: " << b.load\_capacity << " people" << std::endl;

out << "Power: " << b.power << " horsepower" << std::endl;

return out;

}

Queue.h

#pragma once

#include "Header.h"

template <class T>

class Queue

{

public:

Queue(); // Конструктор по умолчанию

bool Empty(); // Возвращает false, если очередь существует

void Add(T& new\_data); // Добавляет элемент в очередь

T Get(); // Возвращает первый элемент и удаляет его из очереди

void Del(); // Удаляет первый элемент из очереди

//void Search(char\* c); // Поиск элемента по имени в очереди и возвращение его порядкового номера

void PrintQueue(); // Выводит на экран объекты, находящиеся в очереди

void NullQueue(); // Удаляет очередь

private:

struct Node

{

T data;

Node\* next;

};

Node\* head, \* tail;

};

Queue.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Queue.h"

#include<iostream>

template <class T>

Queue<T>::Queue() {

head = NULL;

tail = NULL;

}

template <class T>

bool Queue<T>::Empty() {

return head == NULL;

}

template <class T>

void Queue<T>::Add(T& new\_data) {

if (Empty()) {

head = new Node;

head->data = new\_data;

head->next = NULL;

tail = head;

}

else {

tail->next = new Node;

tail = tail->next;

tail->data = new\_data;

tail->next = NULL;

}

}

template <class T>

void Queue<T>::Del() {

if (Empty()) {

std::cout << "Queue is empty" << endl;

}

Node\* tmp = head;

head = head->next;

delete(tmp);

}

template <class T>

void Queue<T>::NullQueue() {

Node\* tmp;

while (!Empty()) {

tmp = head;

head = head->next;

delete(tmp);

}

}

template <class T>

T Queue<T>::Get() {

if (Empty()) {

std::cout << "Queue is empty" << std::endl;

T n;

return n;

}

else {

T r\_data = head->data;

Node\* tmp = head;

head = head->next;

delete tmp;

return r\_data;

}

}

template <class T>

void Queue<T>::PrintQueue() {

if(Empty())

std::cout << "Queue is empty" << std::endl;

Node\* tmp = head;

while (tmp != NULL) {

cout << tmp->data << endl;

tmp = tmp->next;

}

}

Source.cpp

#include <iostream>

#include "Header.h"

#include "Queue.h"

#include "Queue.cpp"

using namespace std;

int main() {

int size, num;

material mat{};

double width;

double length;

double m;

int load\_capacity;

int power;

//Создание объекта obj1

cin >> size;

char\* name = new char[size + 1];

cin >> name;

cin >> num;

switch (num) {

case 1: mat = material::wood; break;

case 2: mat = material::PWH; break;

case 3: mat = material::plastic; break;

}

cin >> width;

cin >> length;

cin >> m;

cin >> load\_capacity;

cin >> power;

boat obj1;

obj1.set\_mat(mat);

obj1.set\_name(name);

obj1.set\_width(width);

obj1.set\_length(length);

obj1.set\_m(m);

obj1.set\_load\_capacity(load\_capacity);

obj1.set\_power(power);

cout << "====================" << endl;

cout << obj1;

cout << "====================" << endl;

delete[] name;

//Создание объекта obj2

cin >> size;

name = new char[size + 1];

cin >> name;

cin >> num;

switch (num) {

case 1: mat = material::wood; break;

case 2: mat = material::PWH; break;

case 3: mat = material::plastic; break;

}

cin >> width;

cin >> length;

cin >> m;

cin >> load\_capacity;

cin >> power;

boat obj2(name, mat, width, length, m, load\_capacity, power);

cout << "====================" << endl;

cout << obj2;

cout << "====================" << endl;

delete[] name;

//Создание объекта obj3

cin >> size;

name = new char[size + 1];

cin >> name;

cin >> num;

switch (num) {

case 1: mat = material::wood; break;

case 2: mat = material::PWH; break;

case 3: mat = material::plastic; break;

}

cin >> width;

cin >> length;

cin >> m;

cin >> load\_capacity;

cin >> power;

boat obj3(obj2);

cout << "====================" << endl;

cout << obj3;

cout << "====================" << endl;

delete[] name;

//Создание объекта obj4

cin >> size;

name = new char[size + 1];

cin >> name;

cin >> num;

switch (num) {

case 1: mat = material::wood; break;

case 2: mat = material::PWH; break;

case 3: mat = material::plastic; break;

}

cin >> width;

cin >> length;

cin >> m;

cin >> load\_capacity;

cin >> power;

boat obj4(mat);

obj4.set\_mat(mat);

obj4.set\_name(name);

obj4.set\_width(width);

obj4.set\_length(length);

obj4.set\_m(m);

obj4.set\_load\_capacity(load\_capacity);

obj4.set\_power(power);

cout << "====================" << endl;

cout << obj4;

cout << "====================" << endl;

delete[] name;

//Вывод объектов

obj1.print();

obj2.print();

obj3.print();

obj4.print();

cout << "Number of objects: " << boat::get\_number\_of\_objects() << endl;

if (obj2 == obj3) {

cout << "==" << endl;

}

else cout << "!=" << endl;

if (obj1 != obj4) {

cout << "!=" << endl;

}

else cout << "==" << endl;

/////////////////////////////////////////////////////////////////////////

Queue<boat> que; // Объявление очереди

// Добавление объектов в очередь

que.Add(obj1);

que.Add(obj2);

que.Add(obj3);

que.Add(obj4);

Queue<int> que1;

int a = 1;

que1.Add(a);

que1.PrintQueue();

Queue<char> que2;

char b = 2;

que2.Add(b);

que2.PrintQueue();

Queue<double> que3;

double c = 3.55555555;

que3.Add(c);

que3.PrintQueue();

Queue<float> que4;

float d = 4.444;

que4.Add(d);

que4.PrintQueue();

que.PrintQueue();// Вывод очереди

char\* tmp\_name = new char[size + 1]; // Создание временной переменной для name

cout << "Enter name: "; cin >> tmp\_name;

//que.Search(tmp\_name); // поиск по имени

delete[] tmp\_name;

cout << "====================" << endl;

que.Del(); //удаление 1 элемента очереди

que.PrintQueue();

cout << "====================" << endl;

que.NullQueue(); // удаление очереди

que.PrintQueue();

cout << "====================" << endl;

system("pause");

return 0;

}

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Для чего используется ключевое слово template?
2. В чем заключаются особенности параметров по умолчанию для шаблонов?
3. Для чего используются шаблоны функций?
4. Назовите разновидности дружественных функций шаблонного класса.
5. Дайте определение специализации шаблона.

1. Для реализации параметризованных типов используется ключевое слово template.

2. Параметры по умолчанию можно инстанцировать при объявлении аргументов, а можно опустить (используются значения по умолчанию).

3. Шаблон функций применяется для создания подходящей функции для любого вызова, однозначно соответствующего его аргументам

4. Дружественные функции могут быть универсальными и могут не использовать спецификацию шаблона (для всех инстанцирований шаблонного класса имеется единственный экземпляр дружественной функции), а также могут быть специфичными и могут задействовать аргументы шаблона для каждого инстанцирования шаблонного класса.

5. Если шаблонный код является неудовлетворительным для конкретного типа аргумента, он может быть специализирован.