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:** | Инкапсуляция; классы; поля и компонентные функции; конструкторы и деструкторы класса |

**Среда разработки:** Microsoft Visual Studio

**Язык программирования:** C++

**Тип проекта:** Консольное приложение

**Задание на лабораторную работу №1**

Череповец

2023

**ТРЕБОВАНИЯ К РАЗРАБОТКЕ**

1. Каждый класс должен быть оформлен в отдельных файлах: заголовочный (.h) и файл с кодом (.cpp).
2. Запрещается использовать обработку исключительных ситуаций и генерировать исключения.
3. Придерживайтесь принципа DRY (Don’t repeat yourself).
4. Обязательно наличие комментариев.

**ЦЕЛЬ РАБОТЫ**

Изучить способы определения классов, правила доступа к элементам; приобрести практические навыки работы с объектами класса; изучить принципы и механизмы создания абстрактных типов данных.

Изучить синтаксис и семантику определения и вызова конструкторов и деструкторов. Приобрести практические навыки создания и уничтожения объектов. Изучить особенности применения различных видов конструкторов.

**ЗАДАНИЯ**

*8 часов*

1. Ознакомьтесь с УМП по ООП часть 1 раздел IV.
2. Разработайте алгоритм и программу, реализующую абстрактный тип данных (АТД) – класс, согласно варианту задания (см. прил. 1):
   1. Предусмотрите закрытую реализацию и открытый интерфейс.
   2. Интерфейс должен содержать функции вывода содержимого и функции доступа к некоторым полям.
   3. Элементы данных должны быть представлены из 5 полей, один из которых должен быть статическим, второй иметь тип указатель, а третий тип перечисление, остальные на свое усмотрение.
   4. Примените основные виды конструкторов и определитесь с деструктором класса.
   5. Проверьте работоспособность АТД на тестовом наборе данных.

**Вариант:** 8. Водный транспорт.

**ПРОГРАММА:**

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(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

~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) {

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

strcpy(name, obj.name);

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(char\* boat\_name) {

if (name) delete[] 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--;

}

Source.cpp

#include <iostream>

#include "Header.h"

using namespace std;

int main() {

boat obj1;

int size, num;

material mat;

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;

}

double width;

cin >> width;

double length;

cin >> length;

double m;

cin >> m;

int load\_capacity;

cin >> load\_capacity;

int power;

cin >> power;

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);

delete[] name;

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);

obj2.set\_mat(mat);

obj2.set\_name(name);

obj2.set\_width(width);

obj2.set\_length(length);

obj2.set\_m(m);

obj2.set\_load\_capacity(load\_capacity);

obj2.set\_power(power);

delete[] name;

cout << obj1.get\_name() << endl;

if (obj1.get\_mat() == material::wood)

cout << "wood" << endl;

if (obj1.get\_mat() == material::PWH)

cout << "PWH" << endl;

if (obj1.get\_mat() == material::plastic)

cout << "plastic" << endl;

cout << obj1.get\_width() << endl;

cout << obj1.get\_length() << endl;

cout << obj1.get\_m() << endl;

cout << obj1.get\_load\_capacity() << endl;

cout << obj1.get\_power() << endl;

cout << obj2.get\_name() << endl;

if (obj2.get\_mat() == material::wood)

cout << "wood" << endl;

if (obj2.get\_mat() == material::PWH)

cout << "PWH" << endl;

if (obj2.get\_mat() == material::plastic)

cout << "plastic" << endl;

cout << obj2.get\_width() << endl;

cout << obj2.get\_length() << endl;

cout << obj2.get\_m() << endl;

cout << obj2.get\_load\_capacity() << endl;

cout << obj2.get\_power() << endl;

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

}

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Класс – это абстрактный тип данных, определяемый пользователем, представляющий собой модель объекта в виде данных и функций для работы с ним. Правила доступа к элементам класса определяются его спецификаторами доступа private и public: элементы, описанные после служебного слова private, видимы только внутри класса, а интерфейс класса (данные, которые видимы не только внутри класса, но и за его пределами) описывается после спецификатора public.
2. Компонентные данные и функции объединяются в классе для того, чтобы сделать код более модульным и повторно используемым и обеспечить возможность инкапсуляции.
3. Доступ к открытым элементам аналогичен доступу к полям структуры: для этого используется операция. При обращении к элементу через имя объекта и операция ->, если объект является указателем. Доступ к закрытым элементам осуществляется посредством вызовов компонентных функций класса, которые располагаются в интерфейсе класса и с этими элементами взаимодействуют.
4. Дружественные функции применяются для доступа к скрытым полям класса и являются альтернативой методам. Методы используются для реализации свойств объекта, а дружественные функции для действий, которые не реализуют свойства объекта, но концептуально входящие в интерфейс класса и нуждающиеся в доступе к его скрытым элементам. Дружественные функции обладают такой же разновидностью, как и обыкновенные функции: они могут быть внешними или компонентными.
5. Под указателем this понимается адрес объекта, который вызывает компонентную функцию. Указатель this в неявном виде используется в методе для ссылок на элементы объекта, т. е. передается в метод по умолчанию, в явном же виде он используется для получения адреса объекта, который вызывает метод.
6. Статические элементы класса описываются с помощью модификатора static. Статические поля класса являются одинаковыми для всех объектов, компилятор выделяет отдельную память в статической области оперативной памяти только один раз, еще до определения объектов. Компонентные статические данные обязательно определяются, подобно компонентным функциям. Для взаимодействия с такими данными необходимо использовать статические компонентные функции.
7. Синтаксис статических компонентных функций выглядит следующим образом:

* Описание: static возвращающий\_тип имя(параметры);
* Определение вне класса:

возвращаемый\_тип имя\_класса::имя\_функции(параметры){тело функции};

* Вызов функции: имя\_класса::имя\_функции(параметры);

Статические компонентные функции обладают той же семантикой, что и обычные методы, но такие функции могут взаимодействовать только со статическими полями класса, а также у них отсутствует неявная передача в качестве параметра указателя this.

Синтаксис константных компонентных функций:

* Описание: возвращающий\_тип имя(параметры)const;
* Определение вне класса:

возвращаемый\_тип имя\_класса::имя\_функции(параметры)const{тело функции};

Особенностью семантики константных компонентных функций является невозможность изменения полей объекта; для константного объекта можно вызывать только константные функции

1. Чтобы изменять компонентные данные объектов, объявленными константами, необходимо наличие ключевого слова mutable при описании поля класса. Это позволит применять компонентные функции, которые смогут изменять значение данных полей.
2. Особенностью создания вложенных классов является место описания класса: private или static, в зависимости от этого вложенный класс будет публичным или закрытым, что определяет его область видимости. Функции вложенного класса могут напрямую ссылаться на статические члены внешнего класса, а также на любые другие типы, определенные во внешнем классе. Доступ к другим членам внешнего класса можно получить из вложенного класса стандартными способами: через объект класса, указатель или ссылку на объект класса. При этом функции вложенного класса могут обращаться в том числе к приватным переменным и константам, которые определены во внешнем классе.