Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Уфимский государственный авиационный технический университет»

Факультет информатики и робототехники

Кафедра Вычислительной математики и кибернетики

**Лабораторная работа №3**

По дисциплине «Объектно-ориентированное программирование»

«Хранилище»

Выполнил: студент группы ПРО-222

Вальшин Руслан Ильдусович

УФА 2020

1. **Цель работы:**

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

1. **Задачи:**

* Функции хранилища объектов
  + добавление объектов
  + удаление объектов
  + переход по объектам (текущий, предыдущий, последующий, проверка наличия)
  + получение очередного объекта из хранилища
  + поочередное обращение к каждому объекту хранилища; вызов функций, реализуемых всеми классами объектов хранилища
* Функции основной программы: код, который в случайном порядке:
  + создаёт объекты
  + добавляет в хранилище
  + использует (для упрощённого вывода)
  + удаляет из хранилища

1. **Ход работы:**

Определение базового класса MaterialObject. У объекта этого класса есть такие свойства как масса (mass) и объём (volume). У класса есть деструктор и 3 конструктора: по умолчанию, с параметрами и копирования. Во всех определениях есть отладочный вывод. Есть виртуальный метод message() для вывода сообщения класса-наследника, если он имеет схожий метод.

class MaterialObject

{

public:

double mass; // В граммах

double volume; // В кубических сантиметрах

MaterialObject()

{

mass = 0;

volume = 0;

printf("Создан объект MaterialObject, mass = %f, volume = %f\n", mass, volume);

}

MaterialObject(double mass, double volume)

{

this->mass = mass;

this->volume = volume;

printf("Создан объект MaterialObject, mass = %f, volume = %f\n", mass, volume);

}

MaterialObject(const MaterialObject& object)

{

mass = object.mass;

volume = object.volume;

printf("Скопирован объект MaterialObject, mass = %f, volume = %f\n", mass, volume);

}

virtual void message()

{

printf("Этот объект принадлежит классу MaterialObject\n");

}

virtual ~MaterialObject()

{

printf("Удаление объекта MaterialObject, mass = %f, volume = %f\n", mass, volume);

}

};

Определение трёх классов-наследников MaterialObject: Food, Clothes и Book. У классов так же есть 3 конструктора и деструктор, метод message() для вывода сообщения класса, и 3 разных собственных методов, которые принимают любое число и выводят соответствующие сообщения.

class Food :public MaterialObject

{

public:

Food()

{

printf("Создан объект Food, mass = %f, volume = %f\n", mass, volume);

}

Food(double mass, double volume)

:MaterialObject(mass, volume)

{

printf("Создан объект Food, mass = %f, volume = %f\n", mass, volume);

}

Food(const Food& object)

:MaterialObject(object)

{

printf("Скопирован объект Food, mass = %f, volume = %f\n", mass, volume);

}

void eatFood(int choice)

{

printf("Съели ");

std::string buffer;

switch (choice % 5)

{

case 0: buffer = "яблоко"; break;

case 1: buffer = "грушу"; break;

case 2: buffer = "банан"; break;

case 3: buffer = "апельсин"; break;

case 4: buffer = "виноград"; break;

default: buffer = "лимон"; break;

}

std::cout << buffer << std::endl;

}

virtual void message()

{

printf("Этот объект принадлежит классу Food\n");

}

virtual ~Food()

{

printf("Удалён объект Food, mass = %f, volume = %f\n", mass, volume);

}

};

class Clothes :public MaterialObject

{

public:

Clothes()

{

printf("Создан объект Clothes, mass = %f, volume = %f\n", mass, volume);

}

Clothes(double mass, double volume)

:MaterialObject(mass, volume)

{

printf("Создан объект Clothes, mass = %f, volume = %f\n", mass, volume);

}

Clothes(const Clothes& object)

:MaterialObject(object)

{

printf("Скопирован объект Clothes, mass = %f, volume = %f\n", mass, volume);

}

void putOnSomeClothes(int choice)

{

printf("Надели ");

std::string buffer;

switch (choice % 5)

{

case 0: buffer = "свитер"; break;

case 1: buffer = "футболку"; break;

case 2: buffer = "джинсы"; break;

case 3: buffer = "носки"; break;

case 4: buffer = "куртку"; break;

default: buffer = "шапку"; break;

}

std::cout << buffer << std::endl;

}

virtual void message()

{

printf("Этот объект принадлежит классу Clothes\n");

}

virtual ~Clothes()

{

printf("Удалён объект Clothes, mass = %f, volume = %f\n", mass, volume);

}

};

class Book :public MaterialObject

{

public:

Book()

{

printf("Создан объект Book, mass = %f, volume = %f\n", mass, volume);

}

Book(double mass, double volume)

:MaterialObject(mass, volume)

{

printf("Создан объект Book, mass = %f, volume = %f\n", mass, volume);

}

Book(const Book& object)

:MaterialObject(object)

{

printf("Скопирован объект Book, mass = %f, volume = %f\n", mass, volume);

}

void readBook(int choice)

{

printf("Прочитали книгу ");

std::string buffer;

switch (choice % 5)

{

case 0: buffer = "Пушкина"; break;

case 1: buffer = "Толстого"; break;

case 2: buffer = "Островского"; break;

case 3: buffer = "Гоголя"; break;

case 4: buffer = "Достоевского"; break;

default: buffer = "Грибоедова"; break;

}

std::cout << buffer << std::endl;

}

virtual void message()

{

printf("Этот объект принадлежит классу Book\n");

}

virtual ~Book()

{

printf("Удалён объект Book, mass = %f, volume = %f\n", mass, volume);

}

};

Определение класса Storage. Класс внутри представляет собой двусвязный список. Структура Node хранит в себе объекты и указатели на следующий и предыдущие элементы. Есть размер хранилища, указатели на первый, последний и текущий элементы. Существует только один конструктор – по умолчанию. Функции:

* add добавляет объект в хранилище
* del – удаляет из хранилища
* previous – возвращает текущий указатель на предыдущий элемент
* next – на следующий элемент
* check – проверяет наличие элемента в хранилище
* getSize – выводит размер хранилище
* getFirst – возвращает указатель на первый объект
* getLast – на последний объект
* getCurrent – на текущий объект
* setFirst – устанавливает текущий указатель на первый элемент
* setLast – на последний элемент
* eol – сообщает о конце списка

class Storage

{

private:

struct Node

{

MaterialObject\* object;

Node\* previous = NULL;

Node\* next = NULL;

};

int size;

Node\* first;

Node\* last;

Node\* current;

public:

Storage()

{

size = 0;

first = NULL;

last = NULL;

current = NULL;

printf("Создан объект Storage, size = 0\n");

}

void add(MaterialObject\* object) // Добавляет объект в хранилище в конец списка

{

Node\* temp = new Node();

temp->object = object;

size++;

if (first == NULL)

{

first = temp;

last = temp;

current = temp;

}

else

{

last->next = temp;

temp->previous = last;

last = temp;

}

}

void del() // Удаляет элемент, на который указывает текущий указатель

{

if (current != NULL)

{

// Переназначение указателей соседних элементов

if(current->previous != NULL)

current->previous->next = current->next;

if(current->next != NULL)

current->next->previous = current->previous;

// Перевод current на следующий или предыдущий элемент

Node\* oldCurrent = current;

if (current->next != NULL)

current = current->next;

else if (current->previous != NULL)

current = current->previous;

else

current = NULL;

// Смена указателей first и last, если current был им равен

if (oldCurrent == first)

first = current;

if (oldCurrent == last)

last = current;

// Удаление элемента из списка

size--;

delete oldCurrent->object;

delete oldCurrent;

}

}

void previous() // Возвращает указатель current на предыдущий элемент в списке, если предыдущий элемент существует

{

if (current != NULL)

if (current->previous != NULL)

current = current->previous;

}

void next() // Возвращает указатель current на следующий элемент в списке, если следующий элемент существует

{

if (current != NULL)

if (current->next != NULL)

current = current->next;

}

bool check(MaterialObject\* object) // Проверяет наличие объекта с тем же указателем в хранилище

{

Node\* buffer = first;

for (int i = 0; i < size; i++, buffer = buffer->next)

if (buffer->object == object)

return true;

return false;

}

int getSize()

{

return size;

}

MaterialObject\* getFirst() // Возвращает ссылку на первый объект в списке

{

return first->object;

}

MaterialObject\* getLast() // Возвращает ссылку на последний объект в списке

{

return last->object;

}

MaterialObject\* getCurrent() // Возвращает ссылку на текущий объект

{

return current->object;

}

void setFirst() // Устанавливает текущий указатель на начало списка

{

current = first;

}

void setLast() // Устанавливает текущий указатель в конец списка

{

current = last;

}

bool eol() // End Of List

{

if (current->next == NULL)

return true;

else

return false;

}

virtual ~Storage()

{

current = first;

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

{

Node\* buffer = current->next;

delete current;

current = buffer;

}

printf("Удалён объект Storage, size = %i\n", size);

}

};

Работа программы. Программа создаёт пустое хранилище. Подряд выполняет 100, 1000 и 10000 случайных действий и выводит время работы каждого блока. В ходе работы программа может: добавить объект в хранилище, удалить из хранилища, вызвать переопределённый виртуальный метод message, вызвать один из собственных методов классов.

int main()

{

setlocale(LC\_ALL, "Russian");

Storage\* box = new Storage();

int count = 10;

while (count < 10000)

{

clock\_t start = clock();

count \*= 10;

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

{

printf("[%i]", i+1);

int key = rand();

if (box->getSize() == 0 || key % 5 <= 1) // Добавить объект

{

if (key % 3 == 0)

box->add(new Food(key % 1000, key % 100));

else if (key % 3 == 1)

box->add(new Clothes(key % 500, key % 200));

else if (key % 3 == 2)

box->add(new Book(key % 500, key % 100));

}

else if (key % 5 == 2) // Удалить объект

{

box->setFirst();

for (int j = 0; j < (key % box->getSize()); j++, box->next())

;

box->del();

}

else if (key % 5 == 3) // Вызов message()

{

box->setFirst();

for (int j = 0; j < (key % box->getSize()); j++, box->next())

;

box->getCurrent()->message();

}

else if (key % 5 == 4) // Вызов личного метода объекта

{

box->setFirst();

for (int j = 0; j < (key % box->getSize()); j++, box->next())

;

MaterialObject\* buffer = box->getCurrent();

Food\* obj1;

Clothes\* obj2;

Book\* obj3;

if (obj1 = dynamic\_cast<Food\*>(buffer))

obj1->eatFood(key);

else if (obj2 = dynamic\_cast<Clothes\*>(buffer))

obj2->putOnSomeClothes(key);

else if (obj3 = dynamic\_cast<Book\*>(buffer))

obj3->readBook(key);

}

}

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("Время работы: %f сек.\n", seconds);

system("pause");

}

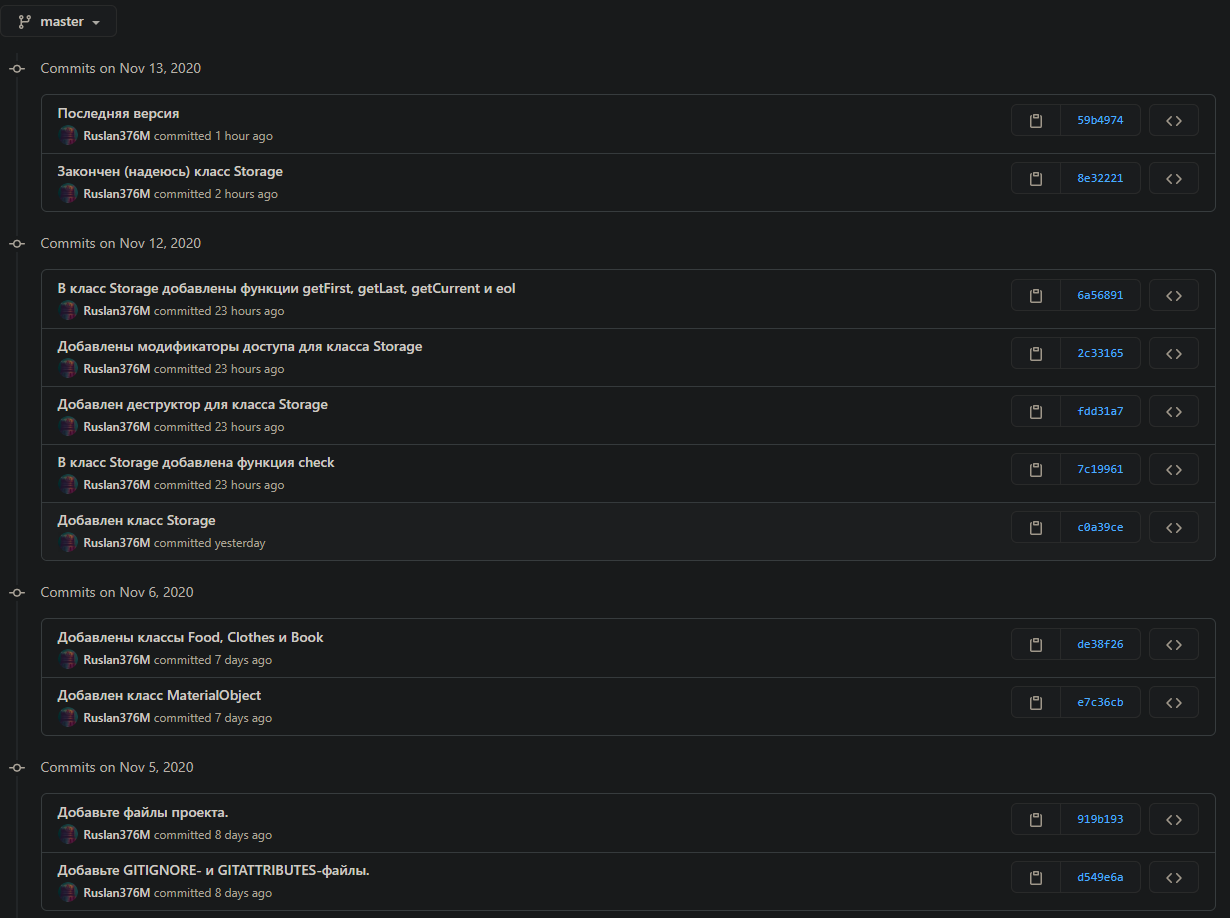
delete box;

system("pause");

}

Приложение 1. Ссылка на GitHub репозиторий и скриншот истории коммитов.

<https://github.com/Ruslan376M/OOP_Lab3>



Приложение 2. Листинг программы.

#include <iostream>

#include <locale.h>

#include <time.h>

class MaterialObject

{

public:

double mass; // В граммах

double volume; // В кубических сантиметрах

MaterialObject()

{

mass = 0;

volume = 0;

printf("Создан объект MaterialObject, mass = %f, volume = %f\n", mass, volume);

}

MaterialObject(double mass, double volume)

{

this->mass = mass;

this->volume = volume;

printf("Создан объект MaterialObject, mass = %f, volume = %f\n", mass, volume);

}

MaterialObject(const MaterialObject& object)

{

mass = object.mass;

volume = object.volume;

printf("Скопирован объект MaterialObject, mass = %f, volume = %f\n", mass, volume);

}

virtual void message()

{

printf("Этот объект принадлежит классу MaterialObject\n");

}

virtual ~MaterialObject()

{

printf("Удаление объекта MaterialObject, mass = %f, volume = %f\n", mass, volume);

}

};

class Food :public MaterialObject

{

public:

Food()

{

printf("Создан объект Food, mass = %f, volume = %f\n", mass, volume);

}

Food(double mass, double volume)

:MaterialObject(mass, volume)

{

printf("Создан объект Food, mass = %f, volume = %f\n", mass, volume);

}

Food(const Food& object)

:MaterialObject(object)

{

printf("Скопирован объект Food, mass = %f, volume = %f\n", mass, volume);

}

void eatFood(int choice)

{

printf("Съели ");

std::string buffer;

switch (choice % 5)

{

case 0: buffer = "яблоко"; break;

case 1: buffer = "грушу"; break;

case 2: buffer = "банан"; break;

case 3: buffer = "апельсин"; break;

case 4: buffer = "виноград"; break;

default: buffer = "лимон"; break;

}

std::cout << buffer << std::endl;

}

virtual void message()

{

printf("Этот объект принадлежит классу Food\n");

}

virtual ~Food()

{

printf("Удалён объект Food, mass = %f, volume = %f\n", mass, volume);

}

};

class Clothes :public MaterialObject

{

public:

Clothes()

{

printf("Создан объект Clothes, mass = %f, volume = %f\n", mass, volume);

}

Clothes(double mass, double volume)

:MaterialObject(mass, volume)

{

printf("Создан объект Clothes, mass = %f, volume = %f\n", mass, volume);

}

Clothes(const Clothes& object)

:MaterialObject(object)

{

printf("Скопирован объект Clothes, mass = %f, volume = %f\n", mass, volume);

}

void putOnSomeClothes(int choice)

{

printf("Надели ");

std::string buffer;

switch (choice % 5)

{

case 0: buffer = "свитер"; break;

case 1: buffer = "футболку"; break;

case 2: buffer = "джинсы"; break;

case 3: buffer = "носки"; break;

case 4: buffer = "куртку"; break;

default: buffer = "шапку"; break;

}

std::cout << buffer << std::endl;

}

virtual void message()

{

printf("Этот объект принадлежит классу Clothes\n");

}

virtual ~Clothes()

{

printf("Удалён объект Clothes, mass = %f, volume = %f\n", mass, volume);

}

};

class Book :public MaterialObject

{

public:

Book()

{

printf("Создан объект Book, mass = %f, volume = %f\n", mass, volume);

}

Book(double mass, double volume)

:MaterialObject(mass, volume)

{

printf("Создан объект Book, mass = %f, volume = %f\n", mass, volume);

}

Book(const Book& object)

:MaterialObject(object)

{

printf("Скопирован объект Book, mass = %f, volume = %f\n", mass, volume);

}

void readBook(int choice)

{

printf("Прочитали книгу ");

std::string buffer;

switch (choice % 5)

{

case 0: buffer = "Пушкина"; break;

case 1: buffer = "Толстого"; break;

case 2: buffer = "Островского"; break;

case 3: buffer = "Гоголя"; break;

case 4: buffer = "Достоевского"; break;

default: buffer = "Грибоедова"; break;

}

std::cout << buffer << std::endl;

}

virtual void message()

{

printf("Этот объект принадлежит классу Book\n");

}

virtual ~Book()

{

printf("Удалён объект Book, mass = %f, volume = %f\n", mass, volume);

}

};

class Storage

{

private:

struct Node

{

MaterialObject\* object;

Node\* previous = NULL;

Node\* next = NULL;

};

int size;

Node\* first;

Node\* last;

Node\* current;

public:

Storage()

{

size = 0;

first = NULL;

last = NULL;

current = NULL;

printf("Создан объект Storage, size = 0\n");

}

void add(MaterialObject\* object) // Добавляет объект в хранилище в конец списка

{

Node\* temp = new Node();

temp->object = object;

size++;

if (first == NULL)

{

first = temp;

last = temp;

current = temp;

}

else

{

last->next = temp;

temp->previous = last;

last = temp;

}

}

void del() // Удаляет элемент, на который указывает текущий указатель

{

if (current != NULL)

{

// Переназначение указателей соседних элементов

if(current->previous != NULL)

current->previous->next = current->next;

if(current->next != NULL)

current->next->previous = current->previous;

// Перевод current на следующий или предыдущий элемент

Node\* oldCurrent = current;

if (current->next != NULL)

current = current->next;

else if (current->previous != NULL)

current = current->previous;

else

current = NULL;

// Смена указателей first и last, если current был им равен

if (oldCurrent == first)

first = current;

if (oldCurrent == last)

last = current;

// Удаление элемента из списка

size--;

delete oldCurrent->object;

delete oldCurrent;

}

}

void previous() // Возвращает указатель current на предыдущий элемент в списке, если предыдущий элемент существует

{

if (current != NULL)

if (current->previous != NULL)

current = current->previous;

}

void next() // Возвращает указатель current на следующий элемент в списке, если следующий элемент существует

{

if (current != NULL)

if (current->next != NULL)

current = current->next;

}

bool check(MaterialObject\* object) // Проверяет наличие объекта с тем же указателем в хранилище

{

Node\* buffer = first;

for (int i = 0; i < size; i++, buffer = buffer->next)

if (buffer->object == object)

return true;

return false;

}

int getSize()

{

return size;

}

MaterialObject\* getFirst() // Возвращает ссылку на первый объект в списке

{

return first->object;

}

MaterialObject\* getLast() // Возвращает ссылку на последний объект в списке

{

return last->object;

}

MaterialObject\* getCurrent() // Возвращает ссылку на текущий объект

{

return current->object;

}

void setFirst() // Устанавливает текущий указатель на начало списка

{

current = first;

}

void setLast() // Устанавливает текущий указатель в конец списка

{

current = last;

}

bool eol() // End Of List

{

if (current->next == NULL)

return true;

else

return false;

}

virtual ~Storage()

{

current = first;

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

{

Node\* buffer = current->next;

delete current;

current = buffer;

}

printf("Удалён объект Storage, size = %i\n", size);

}

};

int main()

{

setlocale(LC\_ALL, "Russian");

Storage\* box = new Storage();

int count = 10;

while (count < 10000)

{

clock\_t start = clock();

count \*= 10;

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

{

printf("[%i]", i+1);

int key = rand();

if (box->getSize() == 0 || key % 5 <= 1) // Добавить объект

{

if (key % 3 == 0)

box->add(new Food(key % 1000, key % 100));

else if (key % 3 == 1)

box->add(new Clothes(key % 500, key % 200));

else if (key % 3 == 2)

box->add(new Book(key % 500, key % 100));

}

else if (key % 5 == 2) // Удалить объект

{

box->setFirst();

for (int j = 0; j < (key % box->getSize()); j++, box->next())

;

box->del();

}

else if (key % 5 == 3) // Вызов message()

{

box->setFirst();

for (int j = 0; j < (key % box->getSize()); j++, box->next())

;

box->getCurrent()->message();

}

else if (key % 5 == 4) // Вызов личного метода объекта

{

box->setFirst();

for (int j = 0; j < (key % box->getSize()); j++, box->next())

;

MaterialObject\* buffer = box->getCurrent();

Food\* obj1;

Clothes\* obj2;

Book\* obj3;

if (obj1 = dynamic\_cast<Food\*>(buffer))

obj1->eatFood(key);

else if (obj2 = dynamic\_cast<Clothes\*>(buffer))

obj2->putOnSomeClothes(key);

else if (obj3 = dynamic\_cast<Book\*>(buffer))

obj3->readBook(key);

}

}

clock\_t end = clock();

double seconds = (double)(end - start) / CLOCKS\_PER\_SEC;

printf("Время работы: %f сек.\n", seconds);

system("pause");

}

delete box;

system("pause");

}