ЗМІСТ

Лабораторна робота № 1.
Конструювання об’єктів конкретних класів

Теоретичні відомості

Для виконання лабораторної роботи необхідне попереднє опрацювання таких тем:

- синтаксис опису класу;
- конструктори і деструктори;
- функції – члени;
- зовнішнє визначення функції – члена;
- метод доступу;
- видимість;
- inline – функції.

Завдання

Побудувати конкретний клас (у відповідності до варіанту), з врахуванням необхідності приховання даних, в якому передбачити: конструктори (в тому числі за замовченням і конструктор копії), деструктор, функції - модифікатори і функції – селектори. Функції – члени мають бути визначені за межами класу.

Варіанти ивідуальних завдань наведено у додатку А.

Звіт

Звіт повинен містити:

1. Титульну сторінку.
2. Завдання на лабораторну роботу.
3. Блок-схему алгоритму розв’язку задачі.
4. Лістинг (роздруківка) програми.
5. Копія екрану з результатами роботи програми.
6. Висновки.

Лабораторна робота № 2.
Реалізація успадкування

Теоретичні відомості

Для виконання лабораторної роботи необхідне попереднє опрацювання таких тем:

- успадкування;
- відкриті і закриті похідні класи;
- ієрархія і композиція;
- правила доступу;
- множинне успадкування;
- порядок виклику конструкторів;
- статичні члени класу.

Завдання

Побудувати похідні класи у відповідності до варіанту. Визначити потрібні статичні члени класу.

Варіанти ивідуальних завдань наведено у додатку А.

Звіт

Звіт повинен містити:

1. Титульну сторінку.
2. Завдання на лабораторну роботу.
3. Блок-схему алгоритму розв’язку задачі.
4. Лістинг (роздруківка) програми.
5. Копія екрану з результатами роботи програми.
6. Висновки.

Лабораторна робота № 3.
Перевантаження операцій

Теоретичні відомості

Для виконання лабораторної роботи необхідне попереднє опрацювання таких тем:

- перевантаження унарних і бінарних операторів;
- перевантаження операцій потокового введення – виведення;
- використання віртуальних функцій;
- дружні класи і дружні функції.

Завдання

В базовому і похідних класах перевантажити операції “=”, потокового введення – виведення як такі, що вводять і виводять повну інформацію про об’єкт в певному форматі, і інші у відповідності до варіанту. Реалізувати вказану віртуальну функцію.

Самостійно продумати і реалізувати спосіб демонстрації отриманих результатів.

Варіанти ивідуальних завдань наведено у додатку А.

Звіт

Звіт повинен містити:

1. Титульну сторінку.
2. Завдання на лабораторну роботу.
3. Блок-схему алгоритму розв’язку задачі.
4. Лістинг (роздруківка) програми.
5. Копія екрану з результатами роботи програми.
6. Висновки.

Лабораторна робота № 4.
Використання можливостей параметричного поліморфізму

Теоретичні відомості

Для виконання лабораторної роботи необхідне попереднє опрацювання таких тем:

- шаблони функцій;
- шаблони класів;
- генерація і опрацювання виключень;

- контейнери, ітератори і алгоритми стандартної бібліотеки STL.

Завдання

У відповідності до варіанту скористатись стандартним контейнером STL або побудувати свій з такою же поведінкою, що і стандартний. Розв'язати задачу з використанням стандартних алгоритмів і операцій контейнерних класів STL.

Варіанти ивідуальних завдань наведено у додатку В.

Приклад виконання лабораторної роботи

```
//  Завдання – відсортувати за збільшенням елементи послідовності з використанням функції
//  порівняння
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <functional>
#include <algorithm>
using namespace std;

//  унарна функція для виведення елементів в cout

template<class T>
class Print: public unary_function<T, void>
{   public:
void operator()(T& arg1)
{   cout<< arg1<<" ";   }   };

// функція порівняння  (використання стандартної операції сортування забезпечує сортування
// за збільшенням, тому в даному випадку необхідно визначити об'єкт порівняння)

bool ReverseCompare(int arg1, int arg2);

// відобразити всі елементи в контейнері

template<class Container>
void ShowElements(Container& c, char* text);

int main(int argc, char* argv[])
{
typedef vector<int> VectorInt;
typedef VectorInt::iterator Itor;

//  створити вектор цілих

VectorInt vInt1(7);
Itor first1 = vInt1.begin();
Itor last1  = vInt1.end();
generate(first1, last1, rand);
ShowElements(vInt1, "Random number sequence");

// відсортувати послідовність за зменшенням

sort(first1, last1, ReverseCompare);
```

```
ShowElements(vInt1, "Sorted in descending order");
return 0;
}
// функція порівняння

bool ReverseCompare(int arg1, int arg2)
{   return (arg1 > arg2);     }

// відобразити всі елементи в контейнері

template<class Container>
void ShowElements(Container& c, char* text)
{ Print<Container::value_type> DoPrint;
cout<<text<<":\n";
for_each(c.begin(), c.end(), DoPrint);
        cout<< "\n\n";     }
```

Звіт

Звіт повинен містити:

1. Титульну сторінку.
2. Завдання на лабораторну роботу.
3. Блок-схему алгоритму розв'язку задачі.
4. Лістинг (роздруківка) програми.
5. Копія екрану з результатами роботи програми.
6. Висновки.

Лабораторна робота №5.
Проектування і програмна реалізація моделі складної системи

Теоретичні відомості

Для виконання лабораторної роботи необхідне попереднє опрацювання таких тем:

- відношення між класами;
- позначення в термінах нотації Буча;
- переведення діаграм класів в С++.

Завдання

Побудувати діаграму класів і діаграму станів і переходів для моделювання роботи системи з певної предметної області у відповідності до варіанту вручну або засобами UML. Скласти програму на С++ на основі побудованих діаграм. Засоби візуалізації роботи системи – довільні.

Варіанти ивідуальних завдань наведено у додатку С.

Звіт

Звіт повинен містити:

1. Титульну сторінку.
2. Завдання на лабораторну роботу.
3. Блок-схему алгоритму розв'язку задачі.

4. Лістинг (роздруківка) програми.
5. Копія екрану з результатами роботи програми.
6. Висновки.

Рекомендована література

1. Б.Страуструп Язык программирования C++, спец.изд. /Пер. с англ — М.; СПб.: «Издательство БИНОМ» – «Невский Диалект», 2002. 1099с.
2. Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес Приемы объектно-ориентированного проектирования. Паттерны проектирования.: "ДМК", "Питер", 2001.
3. Буч Г. Объектно-ориентированное проектирование с примерами применения: Пер. с англ. – М.: Конкорд, 1992. – 519с.,ил.
4. С.Липман, Ж.Лажойе, Язык программирования C++. Вводный курс. 3-е издание./Пер. с англ — М.; СПб.: «Издательство БИНОМ» – «Невский Диалект», 2002. 1104с.
5. Г. Шилдт Теория и практика C++.: "BHV – Санкт-Петербург", 1999 416 с.
6. Скотт Мейерс Эффективное использование C++. 50 рекомендаций по улучшению ваших программ и проектов : "ДМК", 2000, 240 с.
7. Джефф Элджер C++ : библиотека программиста.: "Питер", 2000, 320 с.
8. Вайнер Р., Пинсон Л. C++ изнутри. Пер. с англ.- К.: ДиаСофт, 1993. — 304с.

Додаток А. Варіанти завдань до лабораторних робіт 1, 2 і 3.

Варіант1.

1. Клас “учасник”: прізвище, телефон, адреса.
2. Похідні: “учасник черги на отримання житла” (дата постановки на облік, наявність пільг, порядковий номер в черзі); “учасник виїзної конференції”(чи потребує поселення, тривалість доповіді, час початку роботи конференції, час початку виступу даного доповідача). В базовому і похідних класах визначити функцію *print* для друку прізвища для базового або прізвища і категорії учасника – черги чи конференції - для похідних.
3. В похідних класах перевантажити бінарні операції порівняння (“<“ та “!=“) – за датою поставлення на облік і тривалістю доповіді; унарний “-“ - для зміни пункту про наявність пільги і про потребу в поселенні - на протилежне значення.

Варіант 2.

1. Клас “фігура”: координати на шахівниці, колір. Метод – “хід” на одну позицію в одному з 4 напрямків.
2. Похідні: ”кінь”, “пішак”(порядковий номер, чи своя половина поля), “ферзь” – зі своїми методами “хід” і “удар”.
3. В похідних класах перевантажити бінарний мінус А-В як “А б’є В" і операцію [ ] - за порядковим номером – для виведення координат відповідної фігури. Функцію “хід” перетворити на віртуальну.

Варіант 3.

1. Клас “фігура”: координати на шахівниці, колір. Метод – “хід” – в одному з двох напрямків.
2. Похідні: “шашка” (порядковий номер) і “дамка”, методи -“хід” і “удар”.
3. В похідних класах перевантажити бінарну функцію А/В як “А б’є В” і оператор перетворення типу (із “шашка” в “дамка”). Функцію “хід” перетворити на віртуальну.

Варіант 4.

1. Клас “прямокутник”: координати верхнього лівого і нижнього правого кутів, порядковий номер.
2. Похідні: “ромб” (довжина другої діагоналі) і коло (центр – перші дві координати, діаметр – діагональ прямокутника). В базовому і похідних класах визначити функцію *draw()*.
3. Перевантажити унарні операції - - як зменшення на 1 розміру фігури, бінарну С=А+В – як дублювання в С об’єкта А із збільшенням діагоналі на розмір діагоналі В. Функцію *draw()* перетворити на віртуальну.

Варіант 5.

1. Клас “нота”: назва, октава, тривалість звучання.
2. Похідні: “звук”(частота) і “зображення”(координати на екрані лівого верхнього кута фрагменту нотного стану). В обох класах – порядковий номер ноти, визначити функцію *output* – для кожного класу з різною реалізацією.
3. В похідних класах перевантажити операції постфіксний ++ - для отримання наступної ноти, префіксний ++ - для збільшення тривалості звучання ноти вдвічі. Функцію *output()* перетворити на віртуальну.

Варіант 6.

1. Клас “іграшка”: ціна, назва, кількість на складі.
2. Похідні: “машина”(наявність дистанційного керування, порядковий номер) і “м’яка іграшка”(матеріал, звук). В усіх класах визначити функцію *print* – для кожного класу з різною реалізацією.
3. В похідних класах перевантажити операції “++” – як збільшення кількості на складі; “<” - як порівняння цін. Функцію *print* перетворити на віртуальну.

Варіант 7.

1. Клас “годинник”: стиль відображення (24 чи 12),години, хвилини, секунди. Метод “плюс секунда” – збільшити поточне значення часу на 1 секунду.
2. Похідні: “з прямокутним табло”(координати двох протилежних кутів) і “з круглим циферблатом”(координати центру, радіус). В усіх класах визначити функцію *draw* – для кожного класу з різною реалізацією.
3. В похідних класах перевантажити операції “++” – для зсуву зображення на 1 позицію; бінарний “+” – для збільшення зображення об’єкта - результату. Функцію *draw* перетворити на віртуальну.

Варіант 8.

1. Клас “Товар”: назва, порядковий номер, постачальник, ціна, кількість одиниць.
2. Похідні: “Промисловий товар”(умови транспортування, місце знаходження: на складі, в торговому залі, на вітрині) і “Харчовий продукт”(дата виготовлення, термін зберігання). В усіх класах визначити функцію *alarm()* – для промислового товару з повідомленням із умов транспортування(“не кантовать”, “осторожно!”, ...), або “товар непридатний для споживання” – для харчового, для базового – просто назва товару.
3. В похідних класах перевантажити операції “++” – як збільшення кількості одиниць для харчового і зміна місця знаходження для промислового; “<” – порівняння за терміном зберігання для харчового і за ціною для промислового. Функцію *alarm()* перетворити на віртуальну.

Варіант 9.

1. Клас “Точка на площині”: координати.
2. Похідні: “комплексне число” і “раціональний дріб”. В усіх класах визначити функцію *print* – для друку назви класу, до якого належить об’єкт.
3. В похідних класах перевантажити операції “<” і “+” бінарні і “-” унарний - у відповідності до їх семантики. Функцію *print* перетворити на віртуальну.

Варіант 10.

1. Клас “Точка на площині”: координати.
2. Похідні: “коло”(радіус) і “прямокутник”(координати протилежного кута). В усіх класах визначити функцію *move* – для руху об’єкта на 1 позицію по x і по y.
3. В похідних класах перевантажити операції “++” – як збільшення розміру об’єкта на 1, “<” – за розміром і C=A+B – об’єкт C - “концентричний” відносно A і більший на відповідні розміри об’єкта B. Функцію *move* перетворити на віртуальну.

Варіант 11.

1. Клас “учасник змагань”: країна, вид спорту, назва учасника.
2. Похідні: “футбольна команда” (кількість забитих голів, результат, порядковий номер) і “легкоатлет” (час, час лідера, відставання від лідера, місце у фінальній таблиці). В усіх класах визначити функцію *print* – друк тільки назви учасника або і назви і кількості голів для футбольної команди або часу для легкоатлета.
3. В похідних класах перевантажити операцію “++” – як збільшення на 1 кількості забитих голів або зменшення на 1 місця в фінальній таблиці; бінарний “-” – як

результат конкретної гри: “нічия”, “перемога” або “поразка” в полях “результат”. Для об’єктів легкоатлет А-В щось виконує тільки для ситуації, коли А стає новим лідером, тобто його час менший, ніж час лідера, тоді необхідно змінити відповідні значення для обох об’єктів. Функцію *print* перетворити на віртуальну.

Варіант 12.

1. Клас “довге число”: кількість знаків, основа системи числення.
2. Похідний: “дріб”(кількість знаків у дробовій частині). Добавити поле – максимальна кількість знаків, в обох класах визначити функцію *view* – для кожного класу з різною реалізацією і функцію для переведення в десяткову систему числення.
3. В обох класах перевантажити бінарні операції “+”, “-“ і “<” і унарний “-” – у відповідності до звичної семантики цих операцій і незалежно від різниць в системах числення агентів. Функцію *view* перетворити на віртуальну.

Варіант 13.

1. Клас “Станція”: координати, назва .
2. Похідні: “Радіостанція” (досяжність, вартість ефірного часу, діапазон частот, порядковий номер),.“залізнична станція” (кількість запасних шляхів, тривалість зупинки швидкісних потягів, категорія, порядковий номер), визначити функцію *view* – для кожного класу з різною реалізацією (назва і категорія).
3. В обох класах перевантажити бінарну операцію “+”: результуючий об’єкт для радіостанцій має сумарну досяжність, мінімальну вартість ефірного часу і об’єднання діапазону частот, для залізничних – сумарну кількість запасних шляхів, максимальну категорію і максимальну тривалість зупинки. Унарна “++” – збільшення відповідно вартості ефірного часу і кількості запасних шляхів. Функцію *view* перетворити на віртуальну.

Варіант 14.

1. Клас “ істота ”: координати, вік, назва. Метод *move* – збільшення координат на 1.
2. Утворити ієрархію: “форма існування” – абстрактний клас з чисто віртуальним методом move. Похідні від нього: “істота”, “рослина” (координати), “нерухомий об’єкт”(координати, назва), похідні від істоти: “хижак” (максимальний вік істот цього класу) і “здобич” (максимальний вік істот цього класу).Визначити функцію *move*– для істот збільшення координат, збільшення віку на 1 і якщо вік > за максимальний, то знищення даного об’єкту.
3. Бінарна операція А-В допустима тільки для об’єктів, які знаходяться поруч (у 8 напрямках) і тільки якщо А – хижак, а В – здобич, або А – здобич, а В - рослина, тоді об’єкт В знищується, а об’єкт А пересувається на одну позицію і його вік збільшується на 1. Функцію *move* перетворити на віртуальну.

Варіант 15.

1. Клас “кліматичні умови”: температура. освітленість. вологість, кислотність гранта .
2. Похідні: “кліматичні умови в теплиці”(оптимальні кліматичні умови, допуски), “кліматичні умови на городі” (критичний рівень вологості, критичні рівні кислотності), визначити функцію *show* – для базового – поточний стан, для похідних – виводити тільки ті значення, які перевищують критичні, і величину цього перевищення.
3. В обох класах перевантажити бінарну операцію “= =”, якщо всі параметри обох об’єктів лежать в межах допустимих, або якщо для обох об’єктів є хоч один параметр, що знаходиться за цими межами і унарну - префіксний “++” для збільшення рівня вологості на 1. Функцію *show* перетворити на віртуальну.

Варіант 16.

1. Клас “давач”: поточне значення, максимально і мінімально допустимі, тип (t, p, i, pH), сигнал тривоги.
2. Похідні: “круглий дисплей”(x, y, R), “прямокутний дисплей”(координати протилежних кутів), визначити для базового класу масив максимальних значень за добу для кожного типу. Функція *view* – поточне і максимальне за добу в залежності від типу - для кожного класу з різною реалізацією і функцію set_current_value, яка має змінювати і поточне значення і, в разі необхідності, максимальне за добу.
3. В обох класах перевантажити бінарну операцію “>” за поточними значеннями, якщо давачи одного типу, і унарну “++” – як R++ і x2++, y2++ відповідно. Функцію *view* перетворити на віртуальну.

Варіант 17.

1. Клас “товар на складі”: назва, кількість, місце розташування) .
2. Похідні: “продукт з малим терміном зберігання”(оптимальна температура, дата поставки, термін зберігання), “хімічний елемент”(оптимальна температура. оптимальна вологість, допуски по температурі і вологості), визначити функцію *attention* – для кожного класу з різною реалізацією.
3. В обох класах перевантажити бінарну операцію “<” за датою поставки і за амплітудою критичних температур, і унарну “++” – збільшення кількості товарів відповідного класу. Функцію *attention* перетворити на віртуальну.

Варіант 18.

1. Клас “фраза”: (кількість слів, кількість символів, кількість різних символів.
2. Похідні: “число”(система числення, довжина дробової частини, форма запису(з фіксованою, з плаваючою крапкою), “речення”(кількість символів в алфавіті, чи ігнорувати регістр), визначити функцію *view* – виведення самої фрази, або разом із значенням основи системи числення, або разом із кількістю символів в алфавіті.
3. В обох класах перевантажити бінарні операції “= =” – порівняння відповідно до семантики, “&“ – порозрядне для війкового запису чисел або як по символьний перетин для речень з врахуванням місця розташування символів. Функцію *view* перетворити на віртуальну.

Варіант 19.

1. Клас “Число”: кількість цифр, основа системи числення .
2. Похідні: “ціле”(наявність знакового розряду), “дійсне”(наявність знакового розряду, довжина дробової частини, форма представлення (static)), визначити функцію *print* – для кожного класу з різною реалізацією: просто значення або з вказуванням типу.
3. В обох класах перевантажити бінарну операцію “/”, у відповідності до звичної семантики і унарну “-” – зміна факту наявності знакового розряду. Реалізувати операцію перетворення типу з відкиданням дробової частини (у ціле) і з врахуванням поточної форми представлення дійсних чисел (у дійсне). Функцію *print* перетворити на віртуальну.

Варіант 20.

1. Клас “коло”: x, y, R, ознака візуалізації (чи відображувати на екрані).
2. Похідні: “вписаний многокутник” (кількість сторін), “описаний многокутник” (кількість сторін, колір), ввести порядковий номер фігури до базового класу, визначити функцію *view* – для кожного класу з різною реалізацією.
3. В обох класах перевантажити “++” – збільшення кількості сторін, унарний “-“ – зміна ознаки візуалізації, і операцію перетворення типу: вписаний – описаний многокутник. Функцію *view* перетворити на віртуальну.

Варіант 21.

1. Клас “книга”: код УДК, назва, автор, рік видання, кількість сторінок.
2. Похідні: “Книга в бібліотеці”(інвентарний номер, ознака наявності, кому видана, вартість), “книга в магазині”(відпускна ціна, кількість екземплярів), визначити функцію *view* – для кожного класу з різною реалізацією.
3. В усіх класах перевантажити бінарні операції “= =” – за кодом УДК, “<“ – за вартістю, відпускною ціною або кількістю екземплярів, в залежності від класу і унарну “++” – збільшити кількість екземплярів, а для бібліотечної – якщо книга не видана, то змінити ознаку наявності і в інтерактивному режимі за повнити поле “кому видана”. Функцію *view* перетворити на віртуальну.

Варіант 22.
1. Клас “обладнання”: назва, вартість, дата виготовлення, група (5, 15, 25 – амортизація) і клас “модернізація обладнання” (дата ремонту, вартість ремонту).
2. Похідний: “працююче обладнання”(дата введення в експлуатацію, залишкова вартість). Похідний “модернізоване обладнання”(додаткова вартість), визначити функцію *view* – для назви і вартості – для кожного класу з різною реалізацією.
3. В усіх класах перевантажити операцію “++” – як збільшення дати на рік і з перерахуванням вартості при необхідності, “<“ – порівняння по вартості (або по залишковій вартості). Функцію *view* перетворити на віртуальну.

Варіант 23.
1. Клас “товар”: назва, виробник.
2. Похідний: “виготовлений товар”(дата виготовлення, кількість), похідний від “виготовленого” - “проданий товар”(шифр партії - складається з дати виготовлення і порядкового номера; дата продажу, кількість в партії), визначити функцію *print* – для кожного класу з різною реалізацією (назва і категорія).
3. В обох класах перевантажити бінарну операцію “+” – за кількістю або кількістю в партії, якщо дата виготовлення співпадає, і унарну “++” – за датою виготовлення або датою продажу. Функцію *print* перетворити на віртуальну.

Варіант 24.
1. Клас “банківський рахунок”: назва банку, номер рахунку, МФО.
2. Похідні: “депозитний” (дата відкриття, період, ставка, сума), “розрахунковий” (дата останньої операції, ставка, залишок), визначити функцію *print* – для кожного класу з різною реалізацією (номер і категорія).
3. В обох класах перевантажити бінарну операцію “+” – як переведення коштів з двох рахунків на третій, при цьому перші два рахунки не закриваються, просто суми на них зводяться до 5, і унарну “--” – як знімання відсотків для депозитного і як операція знімання 5 гривень для розрахункового (дата – поточна). Функцію *print* перетворити на віртуальну.

Варіант 25.
1. Клас “підключення”: назва мережі, наявність пільги.
2. Похідні: “за контрактом” (номер рахунку, дата відкриття, кваліфікація рахунку, залишок), “за карткою” (дата закінчення, залишок), визначити функцію *print* – для кожного класу з різною реалізацією (назва мережі і категорія).
3. В обох класах перевантажити бінарну операцію “<” – за залишками, і унарну “--” – як розмову, тривалість якої вводиться інтерактивно, якщо виконані всі потрібні умови, дата – поточна, в результаті об’єкт “за карткою” може бути знищений. Функцію *print* перетворити на віртуальну.

Працездатну версію програми потрібно продемонструвати окремо для кожної лабораторної роботи, вміти відповісти на запитання до відповідної роботи, протокол – на всі три як на єдине ціле.

Додаток Б. Варіанти завдань до лабораторної роботи №4

Варіант 1.

Текст представлений у вигляді вектора, елементи якого – окремі рядки тексту. Відомо, що текст складається з декількох частин, кожна з яких починається словом "Розділ", розташованим в окремому рядку.

Перетворити вказаний текст на стільки окремих списків, скільки розділів в ньому.
Побудувати функції:

- порівняння двох списків.
- вклеювання частини одного списку (починаючи з певного номера n і до кінця) – в другий – перед *i* – м елементом (з вилученням із першого). Передбачити виключні ситуації: *i, n, n+k* – за межами списку.
- визначити, чи відсортовані елементи даних списків, і, якщо відсортовані, злити їх в один.

Варіант 2.

Відсортувати елементи вектора (із варіанта 1) і перетворити на два списки так, щоб в першому були тільки ті елементи, значення яких не повторюється, а в другому – тільки ті, що мають дублікати (але без повторень), наприклад, вектор з елементами *a, b, c, b, d, e, d, d* перетвориться на списки: I - *a, c, e*; II – *b, d.*
Побудувати функції:

- визначення довжини списку.
- перетворення списку в своє дзеркальне відображення.
- виведення на екран *i* – го елементу – передбачити виключну ситуацію, якщо *i* – за межами списку.
- знищення списку з використанням механізму виключної ситуації.

Варіант 3.

Вектор – колода із 36 карт (масть, ранг). Зарезервувати в ньому місце ще для 16 карт додатково. Перетворити вектор в стек, "перетасувавши" спочатку його елементи. Побудувати функції, необхідні для імітації гри в "дурня":

- вибрати 6 карт зверху, утворивши список з відібраних карт.
- відсортувати елементи списку, передбачити свій оператор порівняння (з врахуванням масті).
- вставка елемента у відсортований список, не порушуючи порядок відсортованості. Використати свою функцію порівняння; виключна ситуація – відсутність відсортованості списку.

Варіант 4.

Вектор із комплексних чисел (поля *Re* і *Im*) – всього *n* елементів. Утворити 2 списки: із перших m елементів і із останніх *n-m-1* (останній елемент лишити).

- Відсортувати списки (за збільшенням модуля).
- Злити 2 списки в один, передбачити як виключну ситуацію відсутність відсортованості в одному із списків.
- Вставити в новоутворений список останній елемент із вектора так, щоб не порушити порядку відсортованості елементів списку.

Варіант 5.

Задані 2 списки: I – співробітники всієї організації (згруповані по відділах) і II – співробітники деякого відділу А.

- Знайти перше входження списку співробітників відділу А в загальному списку і циклічно пересунути елементи I списку так, щоб він починався зі списку

співробітників відділу А.

- Виключна ситуація – відсутність входження списку II в список I.
- Відсортувати ту частину загального списку, яка включає співробітників відділу А.

Варіант 6.

Масив містить об'єкти – кістки "доміно" – всього 28 елементів, впорядкований. Перетворити цей масив у список так, щоб окремі елементи були "перетасовані". Створити функції, які були б необхідні для моделювання гри на 2 гравці в доміно. Всі ігрові набори кісток (окремих гравців, "базар", "черга") реалізувати як списки.

Функції:

- вибір необхідного елемента із списку для доповнення черги з однієї із двох сторін, виключні ситуації: – відсутність потрібного елемента, обробник має доповнити список даного гравця елементами із списку "базар"; ситуація "риба" – коли черга з обох сторін в принципі не може бути доповнена тому, що всі кістки з такими значеннями, як у неї на кінцях, уже знаходяться в черзі.
- виведення на екран поточної ігрової ситуації.

Варіант 7.

Імітація черги на отримання житла. Клас "учасник черги" – прізвище, телефон, дата постановки в чергу. Невпорядкований масив "учасників черги" перетворити в стандартний контейнер *priority queue*. Побудувати функції:

- визначення довжини черги;
- додавання нового елемента в чергу (постановка на облік нового учасника);
- вилучення учасників при отриманні житла (в порядку черги).

Передбачити виключні ситуації: дата постановки на облік > поточної (при створенні черги); дата постановки на облік ≠ поточній для нових учасників.

Варіант 8.

Побудувати свій контейнер *uses_priority_queue*, який мав би поведінку, еквівалентну стандартному *priority_queue* із *STL* щодо компонування елементів. Передбачити функцій, аналогічні функціям варіанта 7.

Варіант 9.

Задано список учасників черги на друк видань: автор, назва, об'єм. Перетворити на стандартну приоритетну чергу у відповідності до положення в списку. Побудувати шаблон функції для виведення на екран повної інформації про всіх учасників черги, визначити унарну об'єкт – функцію для виведення одного елемента.

Варіант 10.

Задано масив, елементи якого містять інформацію про підручники в бібліотеці: код УДК, автор, назва, ціна, кількість примірників. Перетворити цей масив на дек. Передбачити функції:

- у відповідності до курсу перевести ціна всіх книжок із гривень у євро. Виключна ситуація: ціна = 0.
- підрахувати кількість книжок одного автора, які є в нашому каталозі. Автор передається як параметр. Виключна ситуація – нульове поле "кількість примірників".
- видалити всю інформацію про книжки певного автора.

Варіант 11.

Реалізувати шаблон контейнерного класу *deque*, який би характеризувався поведінкою, еквівалентному стандартному контейнеру *STL*. Перевірити для об'єктів типу "раціональний

дріб”.

Варіант 12.

Реалізувати шаблон контейнерного класу *stack*, який би характеризувався поведінкою, еквівалентному стандартному контейнеру *STL*. Перевірити для об’єктів типу “гральна карта”.

Варіант 13.

Реалізувати шаблон контейнерного класу *queue*, який би характеризувався поведінкою, еквівалентному стандартному контейнеру *STL*. Перевірити для об’єктів типу “студент”.

Варіант 14.

Реалізувати шаблон контейнерного класу “кільце”, який був би побудований на базі одного із стандартних послідовних контейнерів *STL* і характеризувався такою поведінкою: можливістю циклічного переміщення “голови”, можливістю проходу в двох напрямках, пошуку елементів за значенням, вклеювання і видалення груп елементів, порівняння двох кілець. Виключна ситуація – відсутність посилань на “голову” (у жодного з елементів кільця).

Варіант 15.

Бібліотечний каталог задано у вигляді асоціативного масиву *map* (ключ – автор, значення – назва книги). Визначити функції:

- чи є книги даного автора в каталозі,
- визначити автора потрібної книги, відсутність книги з такою назвою взагалі реалізувати як виключну ситуацію,
- додавання нового елемента в масив,
- виключення елемента за заданим ключем із масиву.

Варіант 16.

Реалізувати шаблон класу *set*, який імітував би роботу з множинами. Передбачити функцію, яка відповідала би логічній операції *in* (*Pascal*), перевантажити операції +, -, * - у відповідності до семантики цих операцій в *Pascal*.

Варіант 17.

Дано вектор об’єктів “раціональний дріб”. Передбачити функції:

- збільшення розмірів вектору на *N* елементів і заповнення значень нових елементів за певним законом (з використанням породжуючої функції), виключна ситуація – відсутність вільної пам’яті потрібного розміру,
- створення нового вектора як копії старого з впорядкуванням перших *n* елементів.

Варіант 18.

Вектор містить коефіцієнти поліному степені *n* - P_n. Побудувати другий вектор – зі значеннями від 0 до 1 з кроком 0.1. Побудувати функції:

- перетворення значень елементів першого вектора на значення відповідного члена полінома для заданого значення x,
- створення третього вектора, елементи якого = $P_n(x_i)$, де x_i – це поточне значення відповідного елемента із другого вектора.

Скористатись можливостями функцій із <*numeric.h*>.

Варіант 19.

Задано два списки. Один складається із елементів “комплексне число” (у вигляді *Re, Im*), другий – із дійсних чисел. Побудувати функції:

- перетворення списку так, щоб кожний елемент містив значення квадрату відповідного числа,

- впорядкування списку за збільшенням.

Обидві функції перевірити для обох списків. Порівняння для комплексних чисел – за модулем.

Варіант 20.

Завдання для варіанту 19. Функції:

- впорядкування тільки тих елементів списку, що задовольняють певній умові (>2 для першого і >2 по модулю для другого),
- розташувати елементи списку в довільному порядку.

Обидві функції перевірити для обох списків. Порівняння для комплексних чисел – за модулем.

Варіант 21.

Задано масив коефіцієнтів поліному так, що *i* – й елемент масиву є коефіцієнтом при x^i. Створити на основі цього масиву стек, який складається із ненульових коефіцієнтів разом зі значенням відповідної степені поліному. Реалізувати функції:

- створення нового поліному як суми двох існуючих (поліном задано у вигляді стеку з відповідними елементами). Ситуацію, коли модуль певного коефіцієнту поліному менший за деяке наперед задане значення ε, розглядати як виключну ситуацію і такий елемент виключити із стеку.
- порівняння двох стеків - поліномів.

Варіант 22.

Побудувати шаблон класу *BTree* – "бінарне дерево", передбачити:

- можливість додавання і видалення певного елемента (за значенням ключа) без перевірки збалансованості дерева, виключна ситуація – відсутність потрібного елемента;
- балансування (за ключем);
- пошук (за ключем).

Варіант 23.

Побудувати шаблон класу *Matrix* – двувимірний масив, для якого передбачити:

- можливість звертання за індексами - як *[i][j]* – без перевірки діапазону;
- можливість звертання за індексами *at(i,j)* – з перевіркою діапазону (аналог функції *at()* для стандартного *vector*), перевірку організувати за допомогою механізму виключних ситуацій;
- можливість збільшення розмірів масиву (аналог *resize* для *vector*).

Варіант 24.

Реалізувати шаблонний клас "динамічний вектор" на основі стандартного контейнера *deque*, в якому передбачити:

- можливість звернутись за індексом *[]* і *at()* – як в стандартному контейнері *vector* - з використанням виключної ситуації для перевірки діапазону;
- функцію для збільшення розміру "динамічного вектора" – аналогічну *resize,* але збільшувати вектор можна з двох боків і виділення динамічної пам'яті реалізувати не так, як в стандартному векторі, а з використанням переваг дека, виключна ситуація – нестача пам'яті.

Варіант 25.

Реалізувати шаблон контейнерного класу *vector*, який би характеризувався поведінкою, еквівалентною стандартному контейнеру *STL*.

Варіант 26.

Реалізувати шаблон контейнерного класу *stack* на базі *string,* який би характеризувався поведінкою, еквівалентною стандартному контейнеру *STL*.

Додаток В. Варіанти завдань до лабораторної роботи №5

Моделювання роботи складної системи. Діюча модель (передбачити асоби візуалізації роботи). Діаграма класів (відповідно до нотації Буча). Діаграма переходів і взаємодій.

1. Імітація роботи ліфта.
2. Імітація роботи музичного центру
3. Імітація роботи систем космічного корабля (виведення на орбіту)
4. Імітація роботи холодильника
5. Імітація роботи пральної машина
6. Імітація роботи охоронної системи приміщення (в тому числі - протипожежна)
7. Імітація роботи телефонного апарату
8. Імітація роботи керованої іграшки автомобіль
9. Імітація роботи керованої іграшки залізниця
10. Імітація роботи керованої іграшки літак
11. Імітація роботи керованої іграшки автозаправна станція
12. Імітація роботи промислової системи водопостачання міста
13. Імітація роботи промислової системи: керування очисними спорудами промислового підприємства
14. Імітація роботи промислової системи: забезпечення безпеки газопроводу
15. Імітація роботи системи забезпечення безпеки руху
16. Імітація роботи автоматичної системи контролю стану систем літака
17. Імітація роботи автоматичної системи контролю стану систем автомобіля
18. Імітація роботи автоматичної системи контролю стану систем підводного човна
19. Імітація роботи банкомата
20. Імітація роботи автомата по продажу цукерок
21. Імітація роботи телевізора
22. Імітація роботи автомату по продажу напоїв
23. Імітація роботи системи опрацювання зображень
24. Імітація роботи системи опрацювання текстів
25. Імітація роботи мікрохвильової печі
26. Імітація роботи турнікета метрополітену
27. Імітація процесу вирощування рослини
28. Імітація роботи системи ППО об’єкту