Міністерство освіти і науки України
Національний університет «Львівська політехніка»
кафедра інформаційних систем та мереж

Григорович Віктор

Об’єктно-орієнтоване програмування

S.O.L.I.D.

Навчальний посібник

2021

Григорович Віктор Геннадійович

Об’єктно-орієнтоване програмування. S.O.L.I.D. Навчальний посібник.

Дисципліна «Об’єктно-орієнтоване програмування» вивчається після курсу «Алгоритмізація та програмування», цією дисципліною продовжується цикл предметів, що стосуються програмування та розробки програмного забезпечення.

В посібнику містяться теоретичні відомості, приклади, методичні вказівки з їх розв’язування, варіанти лабораторних завдань та питання і завдання з контролю знань з теми «S.O.L.I.D.».

Розглядаються наступні роботи лабораторного практикуму:

Лабораторна робота № 8.1.
Індивідуальні та командні проекти

Лабораторна робота № 8.2.
Індивідуальні та командні проекти

Лабораторна робота № 8.3.
Індивідуальні та командні проекти

Відповідальний за випуск – Григорович В.Г.

Стислий зміст

Зміст

Вступ

Дисципліна «Об’єктно-орієнтоване програмування» вивчається після курсу «Алгоритмізація та програмування», цією дисципліною продовжується цикл предметів, що стосуються програмування та розробки програмного забезпечення.

В посібнику містяться теоретичні відомості, приклади, методичні вказівки з їх розв’язування, варіанти лабораторних завдань та питання і завдання з контролю знань з теми «S.O.L.I.D.».

Тема 8. S.O.L.I.D.

Стисло та головне про S.O.L.I.D.

S.O.L.I.D. – це 5 принципів об’єктно-орієнтованого програмування, що описують архітектуру програмного забезпечення. Всі шаблони проектування (патерни) засновані на цих принципах. Абревіатура S.O.L.I.D. розшифровується так:

S (*Single Responsibility Principle*) – принцип єдиної відповідальності (SRP).

O (*Open Closed Principle*) – принцип відкритості / закритості (OCP).

L (*Liskov Substitution Principle*) – принцип підстановки Лісков, що описує можливості замінюваності екземплярів об’єктів (LSP).

I (*Interface Segregation Principle*) – принцип відокремлення інтерйесів (ISP).

D (*Dependency Inversion Principle*) – принцип інверсії залежностей (DIP).

Теоретичні відомості

S.O.L.I.D.

S.O.L.I.D. – це 5 принципів об'єктно-орієнтованого програмування, що описують архітектуру програмного забезпечення. Всі шаблони проектування (патерни) засновані на цих принципах. Абревіатура S.O.L.I.D. розшифровується так:

S (*Single Responsibility Principle*) – принцип єдиної відповідальності (SRP).

O (*Open Closed Principle*) – принцип відкритості / закритості (OCP).

L (*Liskov Substitution Principle*) – принцип підстановки Лісков, що описує можливості замінюваності екземплярів об'єктів (LSP).

I (*Interface Segregation Principle*) – принцип відокремлення інтерйесів (ISP).

D (*Dependency Inversion Principle*) – принцип інверсії залежностей (DIP).

SRP – принцип єдиної відповідальності

Цей принцип означає, що кожний клас чи подібна структура має відповідати тільки за одну мету: всі елементи цього класу мають бути пов'язані однією метою. Клас не має бути схожий на швейцарський ніж, в якому при зміні одного з елементів потрібно змінювати весь інструментарій. Це не означає, що класи мають містити тільки один метод або властивість. Може бути багато елементів, якщо вони відносяться до єдиної відповідальності.

Принцип єдиної відповідальності дає хороший спосіб визначення класів на етапі проектування програми і змушує наперед продумати всі способи зміни класу. Хороший поділ обов'язків виконується лише тоді, коли є повна картина того, як програма має працювати.

Принцип єдиної відповідальності (*Single Responsibility Principle*) можна сформулювати так:

Клас має мати лише однуа причину для зміни.

Під відповідальностю тут розуміється набір функцій, які виконують єдине завдання. Суть цього принципу полягає в тому, що клас має виконувати одну єдину задачу. Весь функціонал класу має бути цілісним, мати високу зв'язність (*high cohesion*).

Конкретне застосування принципу залежить від контексту. В нашому випадку важливо розуміти, як змінюється клас. Якщо клас виконує кілька різних функцій, і вони змінюються окремо, то це якраз той випадок, коли можна застосувати принцип єдиної відповідальності. Тобто іншими словами, клас має кілька причин для зміни.

Але якщо ж усі функції класу, як правило, змінюються разом і становлять одне функціональне ціле, вирішують одну задачу, то немає сенсу застосовувати даний принцип.

Приклад 1

Розглянемо приклад:

```
namespace SOLID
{
    public class Employee
    {
        public int ID { get; set; }

        public string FullName { get; set; }

        /// <summary>
        /// Цей метод добавляє до БД нового працівника
        /// </summary>
        /// <param name="emp">Об'єкт для вставки</param>
        /// <returns>Результат вставки нових даних</returns>
        public bool Add(Employee emp)
        {
            // Вставити дані про працівника у таблицю БД
            return true;
        }

        /// <summary>
        /// Звіт про працівника
        /// </summary>
        public void GenerateReport(Employee em)
        {
            // Генерація звіту про роботу працівника
        }
    }
}
```

В цьому випадку клас `Employee` не відповідає принципу *SRP*, тому що несе дві відповідальності – додавання нового працівника до бази даних і створення звіту. Клас `Employee` не має нести відповідальність за звітність, тому що наприклад, якщо через якийсь час нам скажуть, що потрібно надати звіт в форматі Excel або змінити алгоритм створення звіту, нам потрібно буде відредагувати клас `Employee`.

Згідно *SRP*, один клас має мати одну відповідальність, тому ми мусимо написати окремий клас для генерації звітів:

```
namespace SOLID
{
    public class Employee
    {
        public int ID { get; set; }

        public string FullName { get; set; }

        /// <summary>
        /// Цей метод добавляє до БД нового працівника
        /// </summary>
```

```
        /// <param name="emp">Об'єкт для вставки</param>
        /// <returns>Результат вставки нових даних</returns>
        public bool Add(Employee emp)
        {
            // Вставити дані про працівника у таблицю БД
            return true;
        }
    }

    public class EmployeeReport
    {
        /// <summary>
        /// Звіт про працівника
        /// </summary>
        public void GenerateReport(Employee emp)
        {
            // Генерація звіту про роботу працівника
        }
    }
}
```

Приклад 2

Розглянемо ще один приклад.

Припустимо, нам потрібно визначити клас звіту, в якому ми можемо переміщуватися по сторінках і який можна виводити на друк. На перший погляд, можна було би визначити цей клас таким чином:

```
class Report
{
    public string Text { get; set; }

    public void GoToFirstPage()
    {
        Console.WriteLine("Перехід до першої сторінки");
    }

    public void GoToLastPage()
    {
        Console.WriteLine("Перехід до останньої сторінки");
    }

    public void GoToPage(int pageNumber)
    {
        Console.WriteLine("Перехід до сторінки {0}", pageNumber);
    }

    public void Print()
    {
        Console.WriteLine("Вивід на друк");
        Console.WriteLine(Text);
    }
}
```

Перші три методи відносяться до навігації по звіту і представляють одне єдине функціональне ціле. Від них відрізняється метод `Print()`, який виводить звіт на друк. Що буде, якщо нам знадобиться виводити звіт на консоль або передати його на принтер для

фізичного друку на папері? Або вивести в файл? Чи зберегти в форматі `html`, `txt`, `rtf` і т.д.? Очевидно, що ми можемо для цього змінити належним чином метод `Print()`. Однак ці зміни навряд чи торкнуться інших методів, які відносяться до навігації по сторінках.

Також вірно і зворотне – зміна методів посторінкової навігації навряд чи вплине на можливість виведення тексту звіту на принтер або на консоль. Таким чином, у нас тут простежуються дві причини для зміни, значить, клас `Report` має дві відповідальності (навігація по сторінках – 1 і вивід на друк – 2), і від однієї з них цей клас треба звільнити.

В цьому випадку ми могли б винести функціонал виведення на друк в окремий клас, а потім застосувати агрегацію:

```
interface IPrinter
{
    void Print(string text);
}

class ConsolePrinter : IPrinter
{
    public void Print(string text)
    {
        Console.WriteLine(text);
    }
}

class Report
{
    public string Text { get; set; }

    public void GoToFirstPage()
    {
        Console.WriteLine("Перехід до першої сторінки");
    }

    public void GoToLastPage()
    {
        Console.WriteLine("Перехід до останньої сторінки");
    }

    public void GoToPage(int pageNumber)
    {
        Console.WriteLine("Перехід до сторінки {0}", pageNumber);
    }
    public void Print(IPrinter printer)
    {
        printer.Print(this.Text);
    }
}
```

Тепер об’єкт `Report` отримує посилання на об’єкт `IPrinter`, який використовується для друку, і через метод `Print()` виводиться вміст звіту:

```
IPrinter printer = new ConsolePrinter();
Report report = new Report();
report.Text = "Hello Wolrd";
report.Print(printer);
```

Побічним позитивним ефектом є те, що тепер функціонал друку інкапсулюється в одному місці, і ми зможемо використовувати його повторно для об'єктів інших класів, а не тільки `Report`.

Проте відповідальності в класах не завжди групуються по методам. Цілком можливо, що в одному методі згруповано кілька відповідальностей. Наприклад:

```
class Phone
{
    public string Model { get; set; }
    public int Price { get; set; }
}

class MobileStore
{
    List<Phone> phones = new List<Phone>();

    public void Process()
    {
        Console.WriteLine("Введіть модель:");
        string model = Console.ReadLine();

        Console.WriteLine("Введіть ціну:");
        int price = 0;
        bool result = Int32.TryParse(Console.ReadLine(), out price);

        if (result == false || price <= 0  || String.IsNullOrEmpty(model))
        {
            throw new Exception("Некоректно введені дані");
        }
        else
        {
            phones.Add(new Phone { Model = model, Price = price });
            // сохраняем данные в файл
            using (System.IO.StreamWriter writer =
                new System.IO.StreamWriter("store.txt", true))
            {
                writer.WriteLine(model);
                writer.WriteLine(price);
            }
            Console.WriteLine("Дані успішно опрацьовані");
        }
    }
}
```

Клас `MobileStore` має один єдиний метод `Process()`, однак цей невеликий метод, несе як мінімум чотири відповідальності: (1) введення даних, (2) їх валідація, (3) створення об'єкта `Phone` і (4) збереження. В результаті клас знає абсолютно все: як отримувати дані, як їх перевіряти, як зберігати. При необхідності до нього можна було б додати ще кілька відповідальностей. Такі класи ще називають "божественними" або "класи-боги", бо вони інкапсулюють в собі абсолютно всю функціональність. Подібні класи є одним з поширених анти-патернів, і їх слід намагатися уникати.

Хоча тут досить небагато коду, проте при наступних змінах метод `Process()` може бути сильно роздутий, а функціонал ускладнений і заплутаний.

Тепер змінимо код класу, інкапсулюємо всі відповідальності в окремих класах:

```
class Phone
{
    public string Model { get; set; }
    public int Price { get; set; }
}

class MobileStore
{
    List<Phone> phones = new List<Phone>();

    public IPhoneReader Reader { get; set; }
    public IPhoneBinder Binder { get; set; }
    public IPhoneValidator Validator { get; set; }
    public IPhoneSaver Saver { get; set; }

    public MobileStore(IPhoneReader reader, IPhoneBinder binder,
                       IPhoneValidator validator, IPhoneSaver saver)
    {
        this.Reader = reader;
        this.Binder = binder;
        this.Validator = validator;
        this.Saver = saver;
    }

    public void Process()
    {
        string[] data = Reader.GetInputData();
        Phone phone = Binder.CreatePhone(data);
        if (Validator.IsValid(phone))
        {
            phones.Add(phone);
            Saver.Save(phone, "store.txt");
            Console.WriteLine("Дані успешно опрацьовані");
        }
        else
        {
            Console.WriteLine("Некоректні дані");
        }
    }
}

interface IPhoneReader
{
    string[] GetInputData();
}
class ConsolePhoneReader : IPhoneReader
{
    public string[] GetInputData()
    {
        Console.WriteLine("Введіть модель:");
        string model = Console.ReadLine();
        Console.WriteLine("Введіть ціну:");
        string price = Console.ReadLine();
        return new string[] { model, price };
    }
}
```

```
interface IPhoneBinder
{
    Phone CreatePhone(string[] data);
}

class GeneralPhoneBinder : IPhoneBinder
{
    public Phone CreatePhone(string[] data)
    {
        if(data.Length>=2)
        {
            int price = 0;
            if(Int32.TryParse(data[1], out price))
            {
                return new Phone { Model = data[0], Price = price };
            }
            else
            {
                throw
                    new Exception("Помилка прив'язки моделі Phone. " +
                                  "Некоректні дані для властивості Price");
            }
        }
        else
        {
            throw
                new Exception("Помилка прив'язки моделі Phone. " +
                              "Недостатньо даних для створення моделі");
        }
    }
}

interface IPhoneValidator
{
    bool IsValid(Phone phone);
}

class GeneralPhoneValidator : IPhoneValidator
{
    public bool IsValid(Phone phone)
    {
        if (String.IsNullOrEmpty(phone.Model) || phone.Price <= 0)
            return false;

        return true;
    }
}

interface IPhoneSaver
{
    void Save(Phone phone, string fileName);
}

class TextPhoneSaver : IPhoneSaver
{
    public void Save(Phone phone, string fileName)
    {
        using (System.IO.StreamWriter writer =
            new System.IO.StreamWriter(fileName, true))
        {
            writer.WriteLine(phone.Model);
```

```
            writer.WriteLine(phone.Price);
        }
    }
}
```

Можливе використання класу:

```
MobileStore store = new MobileStore
                    (
                         new ConsolePhoneReader(),
                         new GeneralPhoneBinder(),
                         new GeneralPhoneValidator(),
                         new TextPhoneSaver()
                    );

store.Process();
```

Тепер для кожної відповідальності визначено свій інтерфейс. Конкретні реалізації відповідальностей встновлюються у вигляді інтрефейса в цільовому класі.

При цьому коду стало більше, в зв'язку з чим програма ускладнилася. І, можливо, подібне ускладнення може здатися недоцільним при наявності одного невеликого методу, який необов'язково буде змінюватися. Однак при модифікації стало набагато простіше вводити новий функціонал без зміни існуючого коду. А всі частини методу `Process()`, будучи інкапсульованими у зовнішніх класах, тепер не залежать один від одного і можуть змінюватися самостійно.

Поширені випадки порушення принципу SRP

Часто принцип єдиної відповідальності порушується при змішуванні в одному класі функціональності різних рівнів. Наприклад, клас виконує обчислення і виводить їх користувачеві, тобто поєднує в собі бізнес-логіку і роботу з призначеним для користувача інтерфейсом. Або клас управляє збереженням / отриманням даних і виконанням над ними обчислень, що також небажано. Клас слід застосовувати тільки для однієї задачі – або бізнес-логіка, або обчислення, або робота з даними.

Інший поширений випадок – наявність в класі або його методах абсолютно непов'язаного між собою функціоналу.

OCP – принцип відкритості / закритості

Головною концепцією цього принципу є те, що клас має бути відкритий для розширень, але закритий від модифікацій. Класи та функції мають бути розроблені так, щоб нова функціональність могла бути додана лише при появі нових вимог. «Закрито для модифікації» означає, що ми вже розробили клас, і він пройшов модульне тестування. Ми не повинні міняти його, поки не знайдемо помилок. Це і означає, що клас має бути відкритим лише для розширень і для цього слід використовувати успадкування.

Принцип відкритості / закритості (*Open / Closed Principle*) можна сформулювати так:

Сутності програми мають бути відкриті для розширення, але закриті для зміни.

Суть цього принципу полягає в тому, що система має бути побудована таким чином, що всі її подальші зміни мусять бути реалізовані за допомогою додавання нового коду, а не зміни вже існуючого.

Приклад 1

Розглянемо приклад:

```
public class EmployeeReport
{
    /// <summary>
    /// Тип звіту
    /// </summary>
    public string TypeReport { get; set; }

    /// <summary>
    /// Звіт по працівнику
    /// </summary>
    public void GenerateReport(Employee em)
    {
        if (TypeReport == "CSV")
        {
            // Генерація звіту в форматі CSV
        }

        if (TypeReport == "PDF")
        {
            // Генерація звіту в форматі PDF
        }
    }
}
```

Проблема в цьому класі полягає в тому, що якщо ми захочемо внести новий тип звіту (наприклад, для вивантаження в Excel), нам знадобиться додати нову умову `if`. Але згідно з принципом *OCP*, наш клас має бути закритий від модифікацій і відкритий для розширень.

Нижче показано, як це можна зробити:

```
public class IEmployeeReport
{
    /// <summary>
    /// Метод для створення звіту
    /// </summary>
    public virtual void GenerateReport(Employee em)
    {
        // Базова реалізація, яку неможна модифікувати
    }
}

public class EmployeeCSVReport : IEmployeeReport
{
    public override void GenerateReport(Employee em)
    {
        // Генерація звіту в форматі CSV
    }
}

public class EmployeePDFReport : IEmployeeReport
{
    public override void GenerateReport(Employee em)
    {
        // Генерація звіту в форматі PDF
    }
}
```

Тепер, якщо ми захочемо додати новий тип звіту, слід просто створити новий клас і успадкувати його від `IEmployeeReport`. Таким чином, клас `IEmployeeReport` закритий від модифікацій, але доступний для розширень.

Приклад 2

Розглянемо ще один приклад – клас кухаря:

```
class Cook
{
    public string Name { get; set; }
    public Cook(string name)
    {
        this.Name = name;
    }

    public void MakeDinner()
    {
        Console.WriteLine("Чистимо картоплю");
        Console.WriteLine("Ставимо почищену картоплю на вогонь");
        Console.WriteLine
            ("Зливаємо залишки води, розминаємо варену картоплю в пюре");
        Console.WriteLine("Посипаємо пюре спеціями і зеленню");
        Console.WriteLine("Картопляне пюре готове");
    }
}
```

За допомогою методу `MakeDinner()` будь-який об’єкт цього класу зможе зробити картопляного пюре:

```
Cook bob = new Cook("Bob");
```

```
bob.MakeDinner();
```

Проте одного вміння готувати картопляне пюре для кухаря навряд чи достатньо. Хотілося б, щоб кухар міг приготувати ще щось. І в цьому випадку ми підходимо до необхідності зміни функціоналу класу, а саме методу `MakeDinner()`. Але, відповідно до принципу *OCP*, класи мають бути відкриті для розширення, але закриті для зміни. Тобто, нам треба зробити клас `Cook` відритим для розширення, але при цьому не змінювати.

Для вирішення цього завдання ми можемо скористатися патерном Стратегія. В першу чергу нам треба винести з класу та інкапсулювати всю ту частину, яка представляє змінну поведінку. У нашому випадку це метод `MakeDinner()`. Однак це не завжди буває просто зробити. Можливо, в класі багато методів, але на початковому етапі складно визначити, які з них будуть змінювати свою поведінку і як саме змінювати. В цьому випадку, звичайно, треба аналізувати можливі способи зміни і вже на підставі аналізу робити висновки. Тобто, все, що піддається змінам, виноситься з класу та інкапсулюється ззовні – у зовнішніх сутності.

Отже, змінимо клас `Cook` наступним чином:

```
class Cook
{
    public string Name { get; set; }

    public Cook(string name)
    {
        this.Name = name;
    }

    public void MakeDinner(IMeal meal)
    {
        meal.Make();
    }
}

interface IMeal
{
    void Make();
}


class PotatoMeal : IMeal
{
    public void Make()
    {
        Console.WriteLine("Чистимо картоплю");
        Console.WriteLine("Ставимо почищену картоплю на вогонь");
        Console.WriteLine
            ("Зливаємо залишки води, розминаємо варену картоплю в пюре");
        Console.WriteLine("Посипаємо пюре спеціями і зеленню");
        Console.WriteLine("Картопляне пюре готове");
    }
}
```

```
class SaladMeal : IMeal
{
    public void Make()
    {
        Console.WriteLine("Нарізаємо помідори і огірки");
        Console.WriteLine("Посипаємо зеленню, сіллю і спеціями");
        Console.WriteLine("Поливаємо соняшниковою олією");
        Console.WriteLine("Салат готовий");
    }
}
```

Тепер приготування їжі абстрагировано в інтерфейсі `IMeal`, а конкретні способи приготування визначені в реалізаціях цього інтерфейсу. А клас `Cook` делегує приготування їжі методу `Make()` об'єкта `IMeal`.

Використання класу:

```
Cook bob = new Cook("Bob");
bob.MakeDinner(new PotatoMeal());
Console.WriteLine();
bob.MakeDinner(new SaladMeal());
```

Результати виконання:

```
Чистимо картоплю
Ставимо почищену картоплю на вогонь
Зливаємо залишки води, розминаємо варену картоплю в пюре
Посипаємо пюре спеціями і зеленню
Картопляне пюре готове

Нарізаємо помідори і огірки
Посипаємо зеленню, сіллю і спеціями
Поливаємо соняшниковою олією
Салат готовий
```

Тепер клас `Cook` закритий від змін, зате ми можемо легко розширити його функціональність, визначивши додаткові реалізації інтерфейсу `IMeal`.

Іншим поширеним способом застосування принципу відкритості / закритості є патерн Шаблонний метод. Переробимо попередній приклад за допомогою цього патерну:

```
using System;


abstract class MealBase
{
    public void Make()
    {
        Prepare();
        Cook();
        FinalSteps();
    }
    protected abstract void Prepare();
    protected abstract void Cook();
    protected abstract void FinalSteps();
}
```

```
class PotatoMeal : MealBase
{
    protected override void Cook()
    {
        Console.WriteLine("Ставимо почищену картоплю на вогонь");
        Console.WriteLine("Варимо близько 30 хвилин");
        Console.WriteLine
            ("Зливаємо залишки води, розминаємо варену картоплю в пюре");
    }

    protected override void FinalSteps()
    {
        Console.WriteLine("Посипаємо пюре спеціями і зеленню");
        Console.WriteLine("Картопляне пюре готове");
    }

    protected override void Prepare()
    {
        Console.WriteLine("Чистимо і миємо картоплю ");
    }
}

class SaladMeal : MealBase
{
    protected override void Cook()
    {
        Console.WriteLine("Нарізаємо помідори та огірки");
        Console.WriteLine("Посипаємо зеленню, сіллю і спеціями");
    }

    protected override void FinalSteps()
    {
        Console.WriteLine("Поливаємо соняшниковою олією");
        Console.WriteLine("Салат готовий");
    }

    protected override void Prepare()
    {
        Console.WriteLine("Миємо помідори та огірки");
    }
}
```

Тепер абстрактний клас `MealBase` визначає шаблонний метод `Make()`, окремі частини якого реалізуються класами-нащадками.

Нехай клас `Cook` тепер приймає набір `MealBase` у вигляді меню:

```
class Cook
{
    public string Name { get; set; }

    public Cook(string name, )
    {
        this.Name = name;
    }

    public void MakeDinner(MealBase[] menu)
    {
        foreach (MealBase meal in menu)
            meal.Make();
    }
}
```

В цьому випадку розширення класу знову ж проводиться за допомогою успадкування класів, які визначають необхідний функціонал.

Застосування класів:

```
MealBase[] menu = new MealBase[] { new PotatoMeal(), new SaladMeal() };

Cook bob = new Cook("Bob");
bob.MakeDinner(menu);
```

Результати виконання:

```
Чистимо і миємо картоплю
Ставимо почищену картоплю на вогонь
Варимо близько 30 хвилин
Зливаємо залишки води, розминаємо варену картоплю в пюре
Посипаємо пюре спеціями і зеленню
Картопляне пюре готове

Миємо помідори та огірки
Нарізаємо помідори та огірки
Посипаємо зеленню, сіллю і спеціями
Поливаємо соняшниковою олією
Салат готовий
```

LSP – принцип підстановки Лісков

Цей принцип стверджує, що «наш код має бути таким, що має бути можливість використовувати будь-який похідний клас замість батьківського класу і працювати з цим похідним класом так само, як і з батьківським, без внесення змін». Цей принцип простий, але дуже важливий для розуміння. Похідний клас не має порушувати визначення типу батьківського класу та його поведінку.

Принцип підстановки Лісков (*Liskov Substitution Principle*) є певним керівництвом по створенню ієрархій успадкування. Початкове визначення цього принципу, яке було дано Барбарою Лісков в 1988 році, виглядало наступним чином:

Якщо для кожного об'єкта o1 *типу* S *існує об'єкт* o2 *типу* T*, такий, що для будь-якої програми* P*, визначеної в термінах* T*, поведінка* P *не змінюється при заміні* o2 *на* o1*, то* S *є підтипом* T.

Тобто, іншими словами, клас S може вважатися підкласом T, якщо заміна об'єктів T на об'єкти S не приведе до зміни роботи програми.

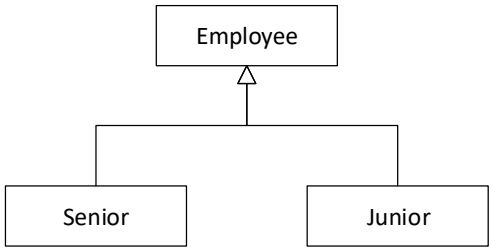
У загальному випадку цей принцип можна сформулювати так:

Має бути можливість замість базового типу підставити будь-який його підтип.

Фактично принцип підстановки Лісков допомагає чіткіше сформулювати ієрархію класів, визначити функціонал для базових і похідних класів і уникнути можливих проблем при застосуванні поліморфізму.

Приклад 1

Розглянемо приклад:



`Employee` – батьківський клас, а `Senior` і `Junior` – дочірні класи, успадковані від `Employee`. Тепер розглянемо код:

```
using System;

namespace SOLID
{
```

```
    public abstract class Employee
    {
        public virtual string GetWorkDetails(int id)
        {
            return "Base Work";
        }

        public virtual string GetEmployeeDetails(int id)
        {
            return "Base Employee";
        }
    }

    public class SeniorEmployee : Employee
    {
        public override string GetWorkDetails(int id)
        {
            return "Senior Work";
        }

        public override string GetEmployeeDetails(int id)
        {
            return "Senior Employee";
        }
    }

    public class JuniorEmployee : Employee
    {
        // Припустимо, для Junior'а відсутні дані
        public override string GetWorkDetails(int id)
        {
            throw new NotImplementedException();
        }

        public override string GetEmployeeDetails(int id)
        {
            return "Junior Employee";
        }
    }
}
```

Здається, що з цим кодом все в порядку. Однак, проаналізуймо наступний код:

```
List<Employee> list = new List<Employee>();

list.Add(new JuniorEmployee());
list.Add(new SeniorEmployee());

foreach (Employee emp in list)
{
    emp.GetEmployeeDetails(985);
}
```

Тепер ми маємо проблему. Для `JuniorEmployee` неможливо отримати інформацію про роботу, тому ми отримаємо неопрацьований виняток, що порушить принцип *LSP*. Для вирішення цієї проблеми слід просто розбити функціонал на два інтерфейси `IWork` та `IEmployee`:

```
public interface IEmployee
{
    string GetEmployeeDetails(int employeeId);
}
```

```
public interface IWork
{
    string GetWorkDetails(int employeeId);
}

public class SeniorEmployee : IWork, IEmployee
{
    public string GetWorkDetails(int employeeId)
    {
        return "Senior Work";
    }

    public string GetEmployeeDetails(int employeeId)
    {
        return "Senior Employee";
    }
}

public class JuniorEmployee : IEmployee
{
    public string GetEmployeeDetails(int employeeId)
    {
        return "Junior Employee";
    }
}
```

Тепер `JuniorEmployee` вимагає реалізації тільки `IEmployee`, а не `IWork`. При такому підході буде підтримуватися принцип *LSP*.

Приклад 2

Проблему, з якою пов'язаний принцип підстановки Лісков, наочно можна продемонструвати на прикладі двох класів: Прямокутник і Квадрат. Нехай вони будуть виглядати наступним чином:

```
class Rectangle
{
    public virtual int Width { get; set; }
    public virtual int Height { get; set; }

    public int GetArea()
    {
        return Width * Height;
    }
}

class Square : Rectangle
{
    public override int Width
    {
        get
        {
            return base.Width;
        }

        set
        {
            base.Width = value;
            base.Height = value;
        }
    }
```

```
    public override int Height
    {
        get
        {
            return base.Height;
        }

        set
        {
            base.Height = value;
            base.Width = value;
        }
    }
}
```

Зазвичай, квадрат представляють як окремий випадок прямокутника – ті ж прямі кути, чотири сторони, тільки ширина обов'язково дорівнює висоті. Тому в класі Квадрат в одній властивості встановлюються відразу і ширина, і висота:

```
set
{
    base.Width = value;
    base.Height = value;
}
```

На перший погляд ніби все правильно, класи надзвичайно прості, всього дві властивості, і, здавалося б, важко десь помилитися. Однак уявімо ситуацію, що в головній програмі у нас наступний код:

```
class Program
{
    static void Main(string[] args)
    {
        Rectangle rect = new Square();
        TestRectangleArea(rect);

        Console.Read();
    }

    public static void TestRectangleArea(Rectangle rect)
    {
        rect.Height = 5;
        rect.Width = 10;
        if (rect.GetArea() != 50)
            throw new Exception("Некоректна площа!");
    }
}
```

З точки зору прямокутника метод `TestRectangleArea()` виглядає нормально, але не з точки зору квадрата. Ми очікуємо, що переданий в метод `TestRectangleArea()` об'єкт буде вести себе як стандартний прямокутник. Однак квадрат, будучи в ієрархії успадкування прямокутником, все ж веде себе не як прямокутник. В результаті програма приведе до винятку.

Іноді для виходу з подібних ситуацій вдаються до спеціального хаку, який полягає в перевірці об’єкта на відповідність типам:

```
public static void TestRectangleArea(Rectangle rect)
{
    if (rect is Square)
    {
        rect.Height = 5;
        if (rect.GetArea() != 25)
            throw new Exception("Неправильна площа!");
    }
    else if (rect is Rectangle)
    {
        rect.Height = 5;
        rect.Width = 10;
        if (rect.GetArea() != 50)
            throw new Exception("Неправильна площа!");
    }
}
```

Але ця перевірка не скасовує того факту, що з архітектурою класів щось не так. Більш того, такі рішення тільки ще більше підкреслюють проблему недосконалості архітектури. Проблема в тому, що похідний клас `Square` не веде себе як базовий клас `Rectangle`, і тому його не слід успадковувати від цього базового класу. В цьому і є практичний сенс принципу Лісков. Похідний клас, який може робити менше, ніж базовий, зазвичай не можна підставити замість базового, і тому він порушує принцип підстановки Лісков.

Існує кілька наборів правил, які мають бути дотримані для виконання принципу підстановки Лісков. Перш за все це правила контракту.

Контракт – це певний інтерфейс базового класу, деякі угоди щодо його використання, яких має дотримуватися клас-нащадок. Контракт задає ряд обмежень або правил, і похідний клас має виконувати ці правила:

- **Передумови** (*Preconditions*) не можуть бути посилені в підкласі. Іншими словами, підкласи не можуть створювати більше передумов, ніж це визначено в базовому класі, для виконання деякої поведінки.

  Передумови представляють набір умов, необхідних для безпомилкового виконання методу. Наприклад:

  ```
  public virtual void SetCapital(int money)
  {
      if (money < 0)
          throw new Exception("Не можна покласти на рахунок менше 0");
      this.Capital = money;
  }
  ```

  Тут умовний вираз служить передумовою – без його виконання не виконуватимуться інші дії, а метод завершиться з помилкою.

Причому об’єктом передумов можуть бути лище загальнодоступні властивості чи поля класу або параметри методу, як в цьому випадку. Приватне поле не може бути об’єктом для передумови, оскільки воно не може бути встановлене ззовні.

Наприклад, в наступному випадку умовний вираз не є передумовою:

```
private bool isValid = false;
public virtual void SetCapital(int money)
{
    if (isValid == false)
        throw new Exception("Валідація не пройдена");
    this.Capital = money;
}
```

Тепер, припустимо, є два класи: `Account` (загальний рахунок) і `MicroAccount` (міні-рахунок з обмеженнями). І другий клас перевизначає метод `SetCapital()`:

```
class Account
{
    public int Capital { get; protected set; }

    public virtual void SetCapital(int money)
    {
        if (money < 0)
            throw new Exception("Не можна покласти на рахунок менше 0");
        this.Capital = money;
    }
}

class MicroAccount : Account
{
    public override void SetCapital(int money)
    {
        if (money < 0)
            throw new Exception("Не можна покласти на рахунок менше 0");

        if (money > 100)
            throw new Exception("Не можна покласти на рахунок більше 100");

        this.Capital = money;
    }
}
```

В цьому випадку підклас `MicroAccount` додає додаткову передумову, тобто підсилює передумови базового класу, що неприпустимо. Тому в реальній задачі ми можемо зіткнутися з проблемою:

```
class Program
{
    static void Main(string[] args)
    {
        Account acc = new MicroAccount();
        InitializeAccount(acc);

        Console.Read();
    }
```

```
    public static void InitializeAccount(Account account)
    {
        account.SetCapital(200);
        Console.WriteLine(account.Capital);
    }
}
```

З точки зору класу `Account` метод `InitializeAccount()` є цілком працездатним. Однак при передачі в нього об'єкта `MicroAccount` ми зіткнемося з помилкою. В результаті пинцип Лісков буде порушений.

- **Післяумови** (*Postconditions*) не можуть бути послаблені в підкласі. Тобто підкласи мають виконувати всі постумови, які визначені в базовому класі.

Післяумови перевіряють стан результуючого об'єкта, що повертається на виході з функції. Наприклад:

```
public static float GetMedium(float[] numbers)
{
    if (numbers.Length == 0)
        throw new Exception("Довжина масиву дорівнює нулю");

    float result = numbers.Sum() / numbers.Length;

    if (result < 0)
        throw new Exception("Результат менше нуля");
    return result;
}
```

Другий умовний вираз тут є післяумовою.

Розглянемо приклад порушення принципу Лісков при ослабленні післяумови:

```
class Account
{
    public virtual decimal GetInterest(decimal sum, int month, int rate)
    {
        // передумова
        if (sum < 0 || month > 12 || month < 1 || rate < 0)
            throw new Exception("Некоректні дані");

        decimal result = sum;
        for (int i = 0; i < month; i++)
            result += result * rate / 100;

        // післяумова
        if (sum >= 1000)
            result += 100; // добавляємо бонус

        return result;
    }
}
```

```
class MicroAccount : Account
{
    public override decimal GetInterest(decimal sum, int month, int rate)
    {
        if (sum < 0 || month > 12 || month < 1 || rate < 0)
            throw new Exception("Некоректні дані");

        decimal result = sum;
        for (int i = 0; i < month; i++)
            result += result * rate / 100;

        return result;
    }
}
```

В якості післяумови в класі `Account` використовується нарахування бонусів в 100 одиниць до фінальної суми, якщо початкова сума – від 1000 і більше. В класі `MicroAccount` ця умова не використовується.

Тепер подивимося, з якою проблемою ми можемо зіткнутися з такими класами:

```
class Program
{
    public static void CalculateInterest(Account account)
    {
        decimal sum = account.GetInterest(1000, 1, 10);
        // 1000 + 1000 * 10 / 100 + 100 (бонус)

        if (sum != 1200) // очікуємо 1200
        {
            throw new Exception("Неочікувана сума при обчисленнях");
        }
    }
    static void Main(string[] args)
    {
        Account acc = new MicroAccount();
        CalculateInterest(acc); // отримуємо 1100 без бонусу 100

        Console.Read();
    }
}
```

Виходячи з логіки класу `Account`, в методі `CalculateInterest()` ми очікуємо отримати в якості результату число `1200`. Однак логіка класу `MicroAccount` показує інший результат. В результаті ми приходимо до порушення принципу Лісков, хоча формально ми просто застосували стандартні принципи ООП – поліморфізм і успадкування.

- **Інваріанти** (*Invariants*) – всі істинні умови базового класу – теж мають бути збережені і в підкласі.

Інваріанти – це деякі умови, які залишаються істинними протягом усього часу існування об’єкта. Як правило, інваріанти передають внутрішній стан об’єкта.

Наприклад:

```
class User
{
    protected int age;

    public User(int age)
    {
        if (age < 0)
            throw new Exception("вік менше нуля");

        this.age = age;
    }

    public int Age
    {
        get { return age; }
        set
        {
            if (value < 0)
                throw new Exception("вік менше нуля");
            this.age = value;
        }
    }
}
```

Поле `age` виступає інваріантом. І оскільки його встановлення можливе лише через конструктор або властивість, то в будь-якому випадку виконання передумови і в конструкторі, і у властивості гарантує, що вік не буде менше 0. І ця обставина збереже свою істинність протягом усього життя об'єкта `User`.

Тепер розглянемо, як тут може бути порушений принцип Лісков. Нехай у нас будуть такі два класи:

```
class Account
{
    protected int capital;

    public Account(int sum)
    {
        if (sum < 100)
            throw new Exception("Некоректна сума");
        this.capital = sum;
    }

    public virtual int Capital
    {
        get { return capital; }
        set
        {
            if (value < 100)
                throw new Exception("Некоректна сума");
            capital = value;
        }
    }
}
```

```
class MicroAccount : Account
{
    public MicroAccount(int sum)
        : base(sum)
    {}

    public override int Capital
    {
        get { return capital; }
        set
        {
            capital = value;
        }
    }
}
```

З точки зору класу `Account` поле не може бути менше 100, і в обох випадках, де йде присвоєння – в конструкторі і у властивості це гарантується. А ось похідний клас `MicroAccount`, перевизначаючи властивість `Capital`, цього вже не гарантує. Тому інваріант класу `Account` порушується.

У всіх трьох перелічених вище випадках проблема вирішується в загальному випадку за допомогою абстрагування і виділення загального функціоналу, який вже успадковують класи `Account` і `MicroAccount`. Тобто, правильна архітектура – коли не один із цих класів успадковується від іншого, а обидва вони успадковуються від одного загального класу.

Таким чином, принцип підстановки Лісков змушує задуматися над правильністю побудови ієрархій класів і застосування поліморфізму, дозволяючи уникнути помилкових ієрархій успадкування і роблячи всю систему класів більш стрункою і несуперечливою.

ISP – принцип відокремлення інтерйесів

Принцип відокремлення інтерфейсів стверджує, що клієнти не мають примусово реалізувати інтерфейси, яких вони не використовують.

Принцип відокремлення інтерфейсів (*Interface Segregation Principle*) відноситься до тих випадків, коли класи мають «жирний інтерфейс», тобто занадто роздутий інтерфейс, не всі методи і властивості якого використовуються і можуть бути затребувані. Таким чином, інтерфейс вийдуть занадто надмірний або «жирним».

Принцип відокремлення інтерфейсів можна сформулювати так:

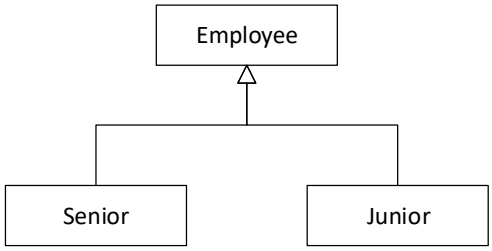
Клієнти не мають вимушено залежати від методів, якими не користуються.

При порушенні цього принципу клієнт, який використовує певний інтерфейс з усіма його методами, залежить від методів, якими не користується, і тому виявляється залежним від змін у цих методах. В результаті ми приходимо до жорсткої залежності між різними частинами інтерфейсу, які можуть бути не пов'язані при його реалізації.

В цьому випадку інтерфейс класу розділяється на окремі частини, які становлять окремі інтерфейси. Потім ці інтерфейси незалежно один від одного можуть застосовуватися і змінюватися. Як наслідок, застосування принципу відокремлення інтерфейсів робить систему слабозв'язною, і тим самим її легше модифікувати та оновлювати.

Приклад 1

Продовжуємо приклад:



`Employee` – батьківський клас, а `Senior` і `Junior` – дочірні класи, успадковані від `Employee`.

Припустимо, що є одна база даних для зберігання даних про працівників всіх типів (тобто, `Junior` і `Senior`). Яким буде кращий підхід для нашого інтерфейсу?

```
public interface IEmployee
{
    bool AddDetailsEmployee();
}
```

Припустимо, що всі класи, похідні від `Employee`, успадковують цей інтерфейс для збереження даних. Тепер припустимо, що замовник одного разу сказав нам, що хоче читати дані тільки для працівників на посаді `Senior`. Що будемо робити, – додамо один метод до цього інтерфейсу?

```
public interface IEmployee
{
    bool AddDetailsEmployee();
    bool ShowDetailsEmployee(int id);
}
```

Але тепер ми щось ламаємо. Ми змушуємо об'єкти `Junior` показувати свої дані з бази даних. Таким чином, рішення полягає в тому, щоб передати цю відповідальність іншому інтерфейсу:

```
public interface IOperationAdd
{
    bool AddDetailsEmployee();
}

public interface IOperationGet
{
    bool ShowDetailsEmployee(int id);
}
```

Тепер клас `Junior` буде реалізувати лише інтерфейс `IOperationAdd`, а `Senior` – обидва інтерфейси. Таким чином забезпечується відокремлення інтерфейсів.

Приклад 2

Припустимо, у нас є інтерфейс відправки повідомлення:

```
interface IMessage
{
    void Send();
    string Text { get; set;}
    string Subject { get; set;}
    string ToAddress { get; set; }
    string FromAddress { get; set; }
}
```

Інтерфейс визначає все основне, що потрібно для відправки повідомлення: саме повідомлення, його тему, адреси відправника та одержувача і, звичайно, сам метод відправки.

Припустимо, є клас `EmailMessage`, який реалізує цей інтерфейс:

```
class EmailMessage : IMessage
{
    public string Subject { get; set; }
    public string Text { get; set; }
    public string FromAddress { get; set; }
    public string ToAddress { get; set; }
```

```
    public void Send()
    {
        Console.WriteLine("Відправляємо по Email повідомлення: {0}", Text);
    }
}
```

Слід відзначити, що клас `EmailMessage` виглядає цілісно, цілком задовольняючи принципу єдиної відповідальності *SPR*. Тобто, з точки зору зв'язності (*cohesion*) тут проблем немає.

Тепер визначимо клас, який би відправляв дані по смс:

```
class SmsMessage : IMessage
{
    public string Text { get; set; }
    public string FromAddress { get; set; }
    public string ToAddress { get; set; }

    public string Subject
    {
        get
        {
            throw new NotImplementedException();
        }

        set
        {
            throw new NotImplementedException();
        }
    }

    public void Send()
    {
        Console.WriteLine("Відправляємо по Sms повідомлення: {0}", Text);
    }
}
```

Тут ми вже стикаємося з невеликою проблемою: властивість `Subject`, яке визначає тему повідомлення, при відправці смс не вказується, тому вона в цьому класі не потрібна. Таким чином, в класі `SmsMessage` з'являється надлишкова функціональність, від якої клас `SmsMessage` починає залежати.

Це не дуже добре, але подивимося далі. Припустимо, нам треба створити клас для відправки голосових повідомлень.

Клас голосової пошти також має відправника і одержувача, тільки саме повідомлення передається у вигляді звуку, що на рівні мови програмування можна виразити у вигляді масиву байтів. І в цьому випадку було б непогано, якби інтерфейс `IMessage` включав би в себе додаткові властивості і методи для цього, наприклад:

```
interface IMessage
{
    void Send();
    string Text { get; set;}
    string ToAddress { get; set; }
```

```
    string Subject { get; set; }
    string FromAddress { get; set; }

    byte[] Voice { get; set; }
}
```

Тоді клас голосової пошти `VoiceMessage` міг би виглядати наступним чином:

```
class VoiceMessage : IMessage
{
    public string ToAddress { get; set; }
    public string FromAddress { get; set; }
    public byte[] Voice { get; set; }

    public string Text
    {
        get
        {
            throw new NotImplementedException();
        }

        set
        {
            throw new NotImplementedException();
        }
    }

    public string Subject
    {
        get
        {
            throw new NotImplementedException();
        }

        set
        {
            throw new NotImplementedException();
        }
    }

    public void Send()
    {
        Console.WriteLine("Передача голосової пошти");
    }
}
```

І тут знову ж таки ми стикаємося з непотрібними властивостями. Плюс нам треба додати нову властивість в попередні класи `SmsMessage` і `EmailMessage`, причому цим класам властивість `Voice` в принципі не потрібна. В результаті тут ми стикаємося з явним порушенням принципу поділу інтерфейсів.

Для вирішення виниклої проблеми нам треба виділити з класів групи пов’язаних методів і властивостей і визначити для кожної групи свій інтерфейс:

```
interface IMessage
{
    void Send();
    string ToAddress { get; set; }
    string FromAddress { get; set; }
}
```

```
interface IVoiceMessage : IMessage
{
    byte[] Voice { get; set; }
}

interface ITextMessage : IMessage
{
    string Text { get; set; }
}

interface IEmailMessage : ITextMessage
{
    string Subject { get; set; }
}

class VoiceMessage : IVoiceMessage
{
    public string ToAddress { get; set; }
    public string FromAddress { get; set; }

    public byte[] Voice { get; set; }
    public void Send()
    {
        Console.WriteLine("Передача голосової пошти");
    }
}

class EmailMessage : IEmailMessage
{
    public string Text { get; set; }
    public string Subject { get; set; }
    public string FromAddress { get; set; }
    public string ToAddress { get; set; }

    public void Send()
    {
        Console.WriteLine("Відправляємо по Email повідомлення: {0}", Text);
    }
}

class SmsMessage : ITextMessage
{
    public string Text { get; set; }
    public string FromAddress { get; set; }
    public string ToAddress { get; set; }
    public void Send()
    {
        Console.WriteLine("Відправляємо по Sms повідомлення: {0}", Text);
    }
}
```

Тепер класи більше не містять методи, які не використовуються. Щоб уникнути дублювання коду, застосовується успадкування інтерфейсів. В результаті структура класів стає простішою, чистішою і яснішою.

Розглянемо ще один приклад:

```
interface IPhone
{
    void Call();
    void TakePhoto();
    void MakeVideo();
    void BrowseInternet();
}

class Phone : IPhone
{
    public void Call()
    {
        Console.WriteLine("Дзвонимо");
    }
    public void TakePhoto()
    {
        Console.WriteLine("Фотографуємо");
    }
    public void MakeVideo()
    {
        Console.WriteLine("Знімаємо відео");
    }
    public void BrowseInternet()
    {
        Console.WriteLine("Серфимо в інтернеті");
    }
}
```

У наш час телефони можуть якщо не все, то дуже багато чого, і представлений вище клас визначає ряд завдань, які виконуються стандартним телефоном: дзвінки, фото, відео, інтернет.

Але нехай у нас є також клас клієнта, який використовує об'єкт `Phone` для фотографування:

```
class Photograph
{
    public void TakePhoto(Phone phone)
    {
        phone.TakePhoto();
    }
}
```

Застосуємо цей клас:

```
Photograph photograph = new Photograph();
Phone lumia950 = new Phone();
photograph.TakePhoto(lumia950);
```

Об'єкт `Photograph`, який представляє фотографа, тепер може фотографувати, використовуючи об'єкт `Phone`.

Проте для фотозйомки можна використовувати також і звичайну фотокамеру, яка не має всіх можливостей телефону. І ми хотіли б, щоб фотограф міг би також використовувати і фотокамеру для фотозйомки. В цьому випадку ми могли взяти загальний інтерфейс `IPhone` і реалізувати його метод `TakePhoto()` в класі фотокамери:

```
class Camera : IPhone
{
    public void Call() { }
    public void TakePhoto()
    {
        Console.WriteLine("Фотографуємо");
    }
    public void MakeVideo() { }
    public void BrowseInternet() { }
}
```

І тут ми знову стикаємося з тим, що клієнт – клас `Camera` залежить від методів, які він не використовує – тобто методів `Call()`, `MakeVideo()`, `BrowseInternet()`.

Для вирішення цієї проблеми ми можемо скористатися принципом відокремлення інтерфейсів:

```
interface ICall
{
    void Call();
}
interface IPhoto
{
    void TakePhoto();
}
interface IVideo
{
    void MakeVideo();
}
interface IWeb
{
    void BrowseInternet();
}

class Camera : IPhoto
{
    public void TakePhoto()
    {
        Console.WriteLine("Фотографуємо за допомогою фотокамери");
    }
}

class Phone : ICall, IPhoto, IVideo, IWeb
{
    public void Call()
    {
        Console.WriteLine("Дзвонимо");
    }
    public void TakePhoto()
    {
        Console.WriteLine("Фотографуємо за допомогою смартфона");
    }
    public void MakeVideo()
    {
        Console.WriteLine("Знімаємо відео");
    }
    public void BrowseInternet()
    {
        Console.WriteLine("Серфимо в інтернеті");
    }
}
```

Для застосування принципу відокремлення інтерфейсів знову ж інтерфейс класу `Phone` розділяється на групи пов'язаних методів (в цьому випадку виходить 4 групи, в кожній по одному методу). Потім кожна група обертається в окремий інтерфейс і використовується самостійно.

За потреби ми можемо застосувати всі інтерфейси відразу, як в класі `Phone`, або лише окремі інтерфейси, як в класі `Camera`.

Тепер змінимо код клієнта – класу фотографа:

```
class Photograph
{
    public void TakePhoto(IPhoto photoMaker)
    {
        photoMaker.TakePhoto();
    }
}
```

Тепер для фотографу не має значення, що передається в метод `TakePhoto()` – фотокамера чи телефон, головне, що цей об'єкт має весь необхідний для фотографування функціонал.

DIP – принцип інверсії залежностей

Цей принцип стверджує, що, по-перше, класи високого рівня не мають залежати від низькорівневих класів. При цьому всі вони мають залежати від абстракцій. По-друге, абстракції не мають залежати від деталей, але деталі мають залежати від абстракцій. Класи високого рівня реалізують бізнес-правила або логіку в системі (додатку). Низькорівневі класи займаються більш докладними операціями, іншими словами, вони можуть займатися записом інформації в базу даних або передачею повідомлень в операційну систему чи служби і т.п.

Кажуть, що високорівневий клас, який має залежність від класів низького рівня або будь-якого іншого класу і багато знає про інші класи, з якими він взаємодіє, тісно з ними пов'язаний. Коли клас явно знає про дизайн та реалізації іншого класу, виникає ризик того, що зміни в одному класі порушать інший клас.

Тому ми повинні тримати такі високорівневі і низькорівневі класи слабо пов'язаними, наскільки це можливо. Щоб це зробити, ми маємо зробити їх залежними від абстракцій, а не один від одного.

Принцип інверсії залежностей (*Dependency Inversion Principle*) служить для створення слабопов'язанних сутностей, які легко тестувати, модифікувати і оновлювати. Цей принцип можна сформулювати наступним чином:

Модулі верхнього рівня не мають залежати від модулів нижнього рівня. І ті й інші мають залежати від абстракцій.

Абстракції не мають залежати від деталей. Деталі мають залежати від абстракцій.

Приклад 1

Розглянемо приклад. Припустимо, що після збереження деяких деталей в базі даних існує одна система повідомлень:

```
public class Email
{
    public void Send()
    {
        // код для відправлення email-листа
    }
}

// Повідомлення
public class Notification
{
    private Email email;
```

```
    public Notification()
    {
        email = new Email();
    }

    public void EmailDistribution()
    {
        email.Send();
    }
}
```

Клас `Notification` повністю залежить від класу `Email`, тому що він відправляє лише один тип повідомлень. А якщо ми захочемо ввести будь-які інші повідомлення, наприклад смс? Тоді нам знадобиться змінити всю систему повідомлень. В цьому випадку це система є тісно пов'язаною. Що ми можемо зробити, щоб вона стала слабо пов'язаною?

Розглянемо наступну реалізацію:

```
public interface IMessenger
{
    void Send();
}

public class Email : IMessenger
{
    public void Send()
    {
        // код для відправлення email-листа
    }
}

public class SMS : IMessenger
{
    public void Send()
    {
        // код для відправлення SMS
    }
}

// Повідомлення
public class Notification
{
    private IMessenger _messenger;

    public Notification()
    {
        _messenger = new Email();
    }

    public void DoNotify()
    {
        _messenger.Send();
    }
}
```

В цьому випадку клас `Notification` все ще залежить від класу `Email`, тому що використовує його об'єкт в конструкторі. В такому випадку ми можемо використовувати принцип впровадження залежностей (*dependency injection* – DI).

Існує три базові принципи впровадження залежностей.

Впровадження залежностей через конструктор (Constructor Injection)

```
public class Notification
{
    private IMessenger _messenger;

    public Notification(IMessenger mes)
    {
        _messenger = mes;
    }

    public void DoNotify()
    {
        _messenger.Send();
    }
}
```

Впровадження залежностей через властивості (Property Injection)

```
public class Notification
{
    private IMessenger _messenger;

    public Notification()
    {}

    public IMessenger Messanger
    {
        set
        {
            _messenger = value;
        }
    }

    public void DoNotify()
    {
        _messenger.Send();
    }
}
```

Впровадження залежностей через метод (Method Injection)

```
public class Notification
{
    public void DoNotify(IMessenger mes)
    {
        mes.Send();
    }
}
```

Приклад 2

Розглянемо наступний приклад:

```
class Book
{
    public string Text { get; set; }
    public ConsolePrinter Printer { get; set; }

    public void Print()
    {
        Printer.Print(Text);
    }
}

class ConsolePrinter
{
    public void Print(string text)
    {
        Console.WriteLine(text);
    }
}
```

Клас `Book`, який представляє книгу, використовує для друку клас `ConsolePrinter`. При такому визначенні клас `Book` залежить від класу `ConsolePrinter`. Більше того, ми жорстко визначили, що друкувати книгу можна лише на консолі за допомогою класу `ConsolePrinter`. Інші ж варіанти, наприклад, вивід на принтер, запис у файл або вивід з використанням якихось елементів графічного інтерфейсу – все це для такої реалізації є неможливим. Абстракція друку книги не відділена від деталей класу `ConsolePrinter`. Таким чином, це є порушенням принципу інверсії залежностей.

Тепер спробуємо привести наші класи у відповідність до принципу інверсії залежностей, відокремивши абстракції від низькорівневої реалізації:

```
interface IPrinter
{
    void Print(string text);
}

class Book
{
    public string Text { get; set; }
    public IPrinter Printer { get; set; }

    public Book(IPrinter printer)
    {
        this.Printer = printer;
    }

    public void Print()
    {
        Printer.Print(Text);
    }
}

class ConsolePrinter : IPrinter
```

```
{
    public void Print(string text)
    {
        Console.WriteLine("Печать на консоли");
    }
}

class HtmlPrinter : IPrinter
{
    public void Print(string text)
    {
        Console.WriteLine("Печать в html");
    }
}
```

Тепер абстракція друку книги відокремлена від конкретних реалізацій. В результаті і клас `Book` і клас `ConsolePrinter` залежать від абстракції `IPrinter`. Крім того, тепер ми можемо створити додаткові низькорівневі реалізації абстракції `IPrinter` і динамічно застосовувати їх в програмі:

```
Book book = new Book(new ConsolePrinter());
book.Print();
book.Printer = new HtmlPrinter();
book.Print();
```

Лабораторний практикум

Оформлення звіту про виконання лабораторних робіт

Вимоги до оформлення звіту про виконання лабораторних робіт №№ 8.1-8.3

Звіт про виконання лабораторних робіт №№ 8.1-8.3 має містити наступні елементи:

1) заголовок;
2) мету роботи;
3) умову завдання;

   Умова завдання має бути вставлена у звіт як фрагмент зображення (скрін) сторінки посібника.
4) UML-діаграму класів;
5) структурну схему програми;

   Структурна схема програми зображує взаємозв'язки програми та всіх її програмних одиниць: схему вкладеності та охоплення підпрограм, програми та модулів; а також схему звертання одних програмних одиниць до інших.
6) текст програми;

   Текст програми має бути правильно відформатований: відступами і порожніми рядками слід відображати логічну структуру програми; програма має містити необхідні коментарі – про призначення підпрограм, змінних та параметрів – якщо їх імена не значущі, та про призначення окремих змістовних фрагментів програми. Текст програми слід подавати моноширинним шрифтом (`Courier New` *розміром 10 пт. або* `Consolas` *розміром 9,5 пт.*) *з одинарним міжрядковим інтервалом;*
7) посилання на git-репозиторій з проектом (див. інструкції з Лабораторної роботи № 2.2 з предмету «Алгоритмізація та програмування»);
8) хоча б для одної функції, яка повертає результат (як результат функції чи як параметр-посилання) – результати unit-тесту: текст програми unit-тесту та скрін результатів її виконання (див. інструкції з Лабораторної роботи № 5.6 з предмету «Алгоритмізація та програмування»);
9) висновки.

Зразок оформлення звіту про виконання лабораторних робіт №№ 8.1–8.3

ЗВІТ

про виконання лабораторної роботи № < *номер* >

« *назва теми лабораторної роботи* »

з дисципліни

«Об’єктно-орієнтоване програмування»

студента(ки) групи КН-26

< *Прізвище Ім’я По_батькові* >

Мета роботи:

…

Умова завдання:

…

UML-діаграма класів:

…

Структурна схема програми:

…

Текст програми:

…

Посилання на git-репозиторій з проектом:

…

Результати unit-тесту:

…

Висновки:

…

Лабораторна робота № 8.1. Індивідуальні та командні проекти

Мета роботи

Навчитися створювати цілісні проекти, використовуючи сучасні підходи до програмування. Для командних проектів – навчитися працювати в команді.

Завдання №1

Реалізувати у вигляді об’єктно-орієнтованої системи структури даних та алгоритми з інших предметів (**Перелік 1**). Реалізовані структури даних мають підтримувати роботу з даними різних типів. Продемонструвати роботу системи на примітивних типах (int, double), бібліотечних типах (string, vector), а також реалізованих класах (**Перелік 2**). Для всіх структур даних реалізувати методи отримання текстового подання (наприклад, з метою виводу на екран), а також генерації відповідних структур даних, заповнених випадковими даними. Можна додатково реалізувати запис відповідних структур у файл та зчитування з файлу (і отримати ***додаткові бали***)

Ця лабораторна робота передбачає використання інтерфейсу командного рядка, проте за бажанням можна реалізувати графічний інтерфейс та/або засоби візуалізації і отримати ***додаткові бали***.

Під час реалізації можна використовувати бібліотеки чи фреймворки, як стандартну бібліотеку C++ (чи іншої обраної мови програмування), так і зовнішні бібліотеки.

Обов’язковою частиною цієї лабораторної є проектування правильного ОО дизайну для поставлених завдань (як зі списку 1, так і зі списку 2). Зокрема, треба дотримуватись основних принципів проектування, уникати дублювання коду. Реалізовані структури даних мають бути зручними для використання – зокрема, клієнт має можливість обирати, яку з реалізацій використовувати, і зміна реалізації не має викликати зміни коду клієнта. Додатково можна реалізувати патерни проектування (і отримати ***додаткові бали***).

Всі реалізовані структури даних та алгоритми мають бути організовані у вигляді однієї програми (один виконуваний файл, одна функція main(), довільна кількість файлів з кодом). За бажання можна розбити на бібліотеку та виконуваний файл, що її використовує. Бажано реалізувати unit tests (і отримати ***додаткові бали***). Якщо тести демонструють всю необхідну поведінку, можна не реалізовувати додаткові режими роботи для перевірки реалізацій (зокрема, інтерактивний режим чи демонстраційний режим). Можна використати

micro-benchmarking frameworks/libraries для виміру часу виконання різних алгоритмів/операцій для різних реалізацій, (і отримати ***додаткові бали***).

В цьому семестрі не обов'язково створювати документацію для реалізованого коду (в наступному семестрі це буде обов'язковою умовою). Проте це бажано зробити, і за це будуть виставлені ***додаткові бали.***

Варіантом цієї лабораторної роботи є номери пунктів зі списків 1 та 2. Різні варіанти мають різний рівень складності, умовно позначений зірочками: * - більш прості завдання, *** - завдання середньої складності, ***** - більш складні завдання. Для отримання кращої оцінки в семестрі варто обирати більш складні варіанти.

Для деяких варіантів запропоновані можливості розвитку завдання, які дозволяють підвищити складність і відповідно отримати кращий результат. Це лише рекомендації, можна зробити якісь інші покращення (хоча бажано погодити з викладачем).

Необхідно вказати в репозиторії обрані варіанти (з обох списків) та реалізовані покращення. Наприклад, в README файлі. Бажано також вказати інші моменти реалізації, за які можуть бути виставлені додаткові бали.

Перелік 1. Структури даних та алгоритми

1. (*) Список, циклічний список. Реалізації на основі зв'язних списків, масивів, бібліотечних засобів. Пошук елемента за значенням, за індексом, першого елемента за заданою умовою.
    a. +* за інші реалізації списку, наприклад, Skip List, XOR Linked List, Unrolled Linked List, VList, Hashed Array Tree, …
2. (**) Список. Реалізації на основі зв'язних списків, масивів, бібліотечних засобів. Алгоритми сортування: insertion sort, quicksort, merge sort.
    a. +* за інші алгоритми сортування;
    b. +* за реалізацію швидких алгоритмів, не заснованих на порівнянні (non-comparison sorts, наприклад counting, radix, bucket sort);
    c. +* за реалізацію загального механізму, що дозволить підключати нові алгоритми сортування;
    d. +* за підтримку різних способів сортування (ключів сортування) для одного об'єкта.
3. (**) Stack, queue, deque (double-ended queue). Реалізації на основі зв'язних списків, масивів, бібліотечних засобів. Стандартні операції для кожної структури. Deque має реалізовувати інтерфейси stack та queue.

4. (***) Розріджені списки та матриці. Доступ за індексом, пошук за значенням, пошук першого елемента за заданою умовою.
    a. +* за операції з матрицями – додавання, множення, множення матриці на вектор, транспонування…
5. (***) Черга з пріоритетом на основі списку (реалізації на основі зв'язного списку та масивів змінної довжини) та дерева. Додавання елементу, забирання елементу (за правилами черги), подивитись наступний елемент, не забираючи його.
    a. +** за реалізацію на основі самобалансованого дерева – AVL, Red-Black, …
    b. +* за реалізацію на основі не-бінарного дерева
    c. +* за реалізацію на основі heap
6. (**) Дерева з довільною кількістю дітей (на основі зв'язного списку та бібліотечних засобів для списку), бінарні дерева, дерева бінарного пошуку. Додавання елементів, видалення елементів (за батьківським вузлом та індексом, за значенням), пошук за значенням.
    a. +* за різні варіанти обходу дерев, пошук за шляхом в дереві;
    b. +* за підтримку різних механізмів видалення, що якось обробляють елементи з видалених під-дерев
7. (****) Графи на основі списку суміжності, матриці суміжності (збереження даних у вершинах та ребрах графів). Додавання та видалення вершин/ребер. Перевірка на зв'язність графу. Визначення відстані між двома вершинами графу.
    a. +* за інші алгоритми на графах
    b. +* за реалізацію загального механізму, що дозволить підключати нові алгоритми
8. (****) Графи, дерева. Додавання та видалення елементів/вершин/ребер. Побудова кістякового дерева для заданого графу.
    a. +* за побудову мінімального кістякового дерева;
    b. +* за кілька різних алгоритмів побудови кістякового дерева і механізм додавання нових алгоритмів;
    c. +* за більш складні алгоритми побудови кістякового дерева.

Перелік 2. Класи для опису даних

1. (*) Фігури на площині: трикутники, чотирикутники, п'ятикутники. Обчислення периметра та площі фігури. Перевірка на спеціальні фігури – прямокутні, рівнобедрені, рівносторонні трикутники; трапеції, паралелограми, ромби, прямокутники, квадрати; правильні п'ятикутники.

a. +* за інші спеціальні типи фігур;
b. +* за побудову спеціальних точок для даної фігури та фігур на їх основі – наприклад, точки перетину медіан/бісектрис/висот, центр описаного кола, точка перетину діагоналей, теорема Наполеона, … –та перевірку відповідних властивостей та теорем геометрії.

2. (***) Фігури на площині: прямі та кола. Обчислення точок перетину двох фігур. Операції симетричного відображення відносно заданої прямої та інверсії відносно заданого кола (для точок та цих фігур).
    a. +* обчислення кутів між фігурами, перевірка збереження кутів під час перетворень
3. (****) Фігури на площині, задаються довільною невід'ємною функцією (фігура від осі x до графіку функції) або двома функціями (фігура між графіками функцій). Обчислення площі та периметру фігур. Обчислення фігури-перетину двох заданих фігур.
    a. +* обчислення фігури – об'єднання двох заданих фігур)
    b. +* за можливість вводити складні функції з клавіатури/через інтерфейс користувача
    c. +* за підтримку різних способів задання функції (символьний, табличний, графічний, код, …)
4. (*****) Тіла в просторі, задаються довільною невід'ємною функцією двох аргументів (тіло від площини xy до графіку функції) або тіла обертання – задаються довільною невід'ємною функцією одного аргументу. Обчислення об'єму та площі поверхні тіла.
    a. +* тіла обертання навколо різних осей
    b. +* за можливість вводити складні функції з клавіатури/через інтерфейс користувача
    c. +* за підтримку різних способів задання функції (символьний, табличний, графічний, код, …)
5. (**) Функції багатьох змінних (підтримуються стандартні функції – арифметичні операції, піднесення до степеня, логарифми, тригонометричні функції). Диференціювання за заданою змінною.
    a. +* спрощення отриманих виразів
6. (***) «Нечесні» гральні кості з кількістю граней N=2, 4, 6, 8, 10, 12, 20. Грані помічено числами від 1 до N. Для кожної грані задається ймовірність випадіння (число від 0 до 1). Обчислення всіх можливих сум для заданого набору костей (набір

може містити довільну кількість костей з однаковими чи різними гранями та однаковими чи різними ймовірностями) та ймовірність випадіння кожної з них.

   a. +* порівняння двох наборів за очікуваною сумою значень граней.

7. (**) Адреси IPv4 (a1.a2.a3.a4, 0<=ai<=255) та IPv6 (b1:b2:b3:b4:b5:b6:b7:b8, кожна з bi – набір від 1 до 4 шістнадцяткових цифр 0..9a..f). Адреси підмереж у форматі CIDR: address/subnet_bits, де address – адреса в одному з форматів, subnet_bits – кількість бітів маски підмережі. Перевірка належності адреси до підмережі.
   a. +* за реалізацію пошуку вільного діапазону адрес заданого розміру (підмережі);
   b. +* за підтримку вкладених підмереж;
   c. +* за підтримку інших варіантів запису IPv6 адрес, наприклад скорочень :: та IPv4 адрес в нотації IPv6;
   d. +* за побудову адрес з текстового подання (parsing);
   e. +* за додаткову підтримку MAC адрес та побудову IPv6 адреси на основі MAC адреси.
8. (*) Інформація про об’єкти файлової системи – файли та каталоги (назва, розмір, час створення, час модифікації, тип файлу). Розрахунок повного шляху до файлу/каталогу. Пошук файлів за заданими критеріями в дереві підкаталогу.
   a. +* за підтримку критеріїв за кількома параметрами одночасно;
   b. +* за підтримку складних критеріїв – довільних виразів з логічними операторами;
   c. +* за пошук імен та шляхів з використанням регулярних виразів (regular expressions);
   d. +* за підтримку symlinks/shortcuts;
   e. +* за реалізацію системи прав доступу – можна задати права доступу до певного файлу чи каталогу, права доступу з каталогу за замовчуванням застосовуються до всіх дітей;
   f. +* за імпорт з реальної файлової системи.
9. (**)Інформація про дату (рік, місяць, день) та час (години, хвилини, секунди). Перевірка правильності дати та часу відповідно до григоріанського календаря. Арифметика моментів часу: різниця між двома моментами часу у заданих одиницях, додавання чи віднімання такої різниці до заданого моменту часу. Обчислення дня тижня для заданої дати.
   a. +* за обчислення додаткових параметрів, наприклад, номер тижня в місяці та в році

b. +* за альтернативні варіанти побудови дати та часу (наприклад, «перший вівторок листопада»)
c. +* за обчислення статистики, наприклад, на який день тижня найчастіше припадає 13 число (за даним діапазоном дат чи взагалі в календарі).
d. +* за підтримку альтернативних календарів (юліанського, …)

10. (***) Інформація про книги (назва, автор(и), дата виходу, кількість сторінок, коротка анотація) та персонажів книг (список імен/псевдонімів, в яких книгах згадується, для кожної книги рівень участі – головний, чи другорядний, чи епізодичний персонаж). Розбиття книг на серії (дві книги належать до однієї серії, якщо у них є спільний головний чи другорядний персонаж; в рамках серії книги впорядковано за датою виходу).
    a. +* за реалізацію «бази даних» - з можливістю додавати, редагувати, видаляти, шукати книги та авторів (аналогічно лабораторній роботі №1 з минулого семестру)
    b. +* за підтримку «альтернативних реальностей» - наприклад, канонічні серії книг, книги на основі фільмів, неофіційні книги, …

Завдання №2

Реалізувати програму з (псевдо)графічним інтерфейсом користувача. Для реалізації інтерфейсу можна використовувати Qt або інший фреймворк (за бажанням студента, погодивши з викладачем).

Далі наведені можливі програми для реалізації, для кожної з яких є основні можливості (які треба обов’язково реалізувати) та список додаткових можливостей (з них треба обрати декілька для реалізації, хоча за бажання можна реалізувати й усі, отримавши за це більше балів). Варіантом цієї лабораторної роботи є номер програми та номери додаткових можливостей, які будуть реалізовані.

Програма 1 – розумний таймер

Базові можливості:

- Можливість запуску таймерів, звуковий та візуальний сигнал по завершенню часу.
- Таймери на певний проміжок часу (власне таймери) або до заданого моменту часу (будильники) – бажано реалізувати як один список, можливо з додатковою фільтрацією
- Можливість одночасного запуску декількох (довільної кількості) таймерів,

- Зручний інтерфейс для перегляду списку таймерів, керування таймерами

Додаткові можливості:

1. Фільтри для вибору таймерів за різними параметрами.
2. Таймери різних типів – дозволяють налаштовувати сигнал, значення часу, інші параметри.
3. Різні формати задання часу таймерів – час від даного моменту, час спрацювання, різні формати часу, інші можливості
4. Підказки під час запуску таймерів – можливість задати певні залежності між таймерами, наприклад, після спрацювання таймера типу 1 рекомендовано запустити таймер типу 2
5. Завчасно налаштовані групи таймерів з можливість швидкого запуску всіх таймерів групи
6. Можливість режиму “Do not disturb” – певного часу, коли сигнали таймерів не спрацьовують.
7. Можливість запам’ятовувати запущені таймери і продовжувати роботу після перезапуску застосунку чи ОС.

Програма 2 – система для нотаток

Базові можливості:

- Можливість швидкого створення текстових нотаток (ідей, конспектів якихось ресурсів, завдань/TODO, …)
- Можливість перегляду створених нотаток в хронологічному порядку (за часом створення)
- Можливість переносити менш потрібні/менш актуальні/виконані нотатки до архіву

Додаткові можливості:

1. Підтримка груп чи контекстів для нотаток – користувач задає, в якому контексті робляться нотатки (наприклад, робочі, персональні, …)
2. Підтримка ієрархічних контекстів (робочі – проект1 – компонент А – …)
3. Можливість зручного перемикання контекстів – список нещодавніх контекстів, пов’язані контексти
4. Можливість задавати ключові слова, які автоматично перемикають на відповідний контекст
5. Global hotkey – можливість додати нотатки натисканням спеціальної комбінації клавіш, з можливістю вставити текст з буфера обміну.

6. Не лише текстові нотатки – зображення, посилання, аудіо, відео, …
7. Можливість експорту певного набору нотаток в один документ.

Програма 3 – offline file manager

Базові можливості:

- Можливість зберігання інформації про файли та каталоги на зовнішніх носіях (флеш-диски, зовнішні диски, CD/DVD)
- Можливість додавати метадані до файлів/каталогів
- Якщо носій підключено – можливість відкрити системний переглядач файлів для заданого файлу чи каталогу.

Додаткові можливості:

1. Можливість створення віртуальних каталогів, які об’єднують дані з різних джерел
2. Пошук за метаданими (ім’я файлу/каталогу, тип, час створення/редагування, хеш/контрольна сума, користувацькі метадані)
3. Інформація про кількість точних копій файлу чи каталогу на відомих носіях
4. Генерація і запуск команд для копіювання чи синхронізації даних між різними носіями (наприклад, з використанням rsync, robocopy чи інших подібних засобів)

Програма 4 – демонстрація роботи алгоритмів чи структур даних

Базові можливості:

- Графічна ілюстрація певної структури даних (де зберігаються дані, куди показують вказівники, …)
- Покрокова ілюстрація виконання алгоритму – як змінюється стан структури даних та додаткових змінних на кожному кроці алгоритму
- Можливість рухатись як вперед, так і назад по крокам алгоритму
- Список підтримуваних алгоритмів/структур даних, можливість вибору зі списку для перегляду демонстрації.

Додаткові можливості:

1. Можливість користувачу задавати параметри структури даних – наприклад, кількість елементів, значення елементів
2. Можливість збереження та відновлення проміжного стану демонстрації.

3. Можливість вибору якихось елементів демонстрації для відслідковування – вони виділяються, наприклад кольором чи шрифтом, відображається інформація про їх зміни.
4. Можливість підключення нових демонстрацій без перекомпіляції основної програми.

Програма 5 – демонстрація випадкових подій

Базові можливості:

- Можливість задати кілька випадкових подій – задаються можливі результати та ймовірності кожного результату.
- Запуск моделювання подій заданої кількості разів
- Список результатів моделювання – які події стались на кожному кроці.

Додаткові можливості:

1. Підтримка інших видів випадкових подій – вибір елементів зі списку без повернення («колода карт»), випадкові блукання по прямій/площині/графу, випадкові точки в множинах на площині, … (як джерело інших ідей щодо можливих типів подій, можна подивитись https://www.random.org/)
2. Збір певної статистики – обчислення середніх значень, кількості подій з певної категорії, …
3. Можливість задати та зберегти seed value генератора випадкових чисел, щоб відтворити ту саму ситуацію.
4. Можливість підключення нових демонстрацій без перекомпіляції основної програми.

Програма 6 – перевірка завдань з програмування

Базові можливості:

- Можливість задавати кілька джерел коду – репозиторії, онлайн файлові сховища, локальна файлова система.
- Прив’язка коду до студентів.
- Можливість додавати коментарі, прив’язані до конкретного коду
- Генерація звітів з підсумками коментарів

Додаткові можливості:

1. Копіювання коду для запуску у спеціально налаштовані середовища
2. Перевірка виконання тестів
3. Перевірка на заданих вхідних даних

4. Запуск статичних аналізаторів коду і збір інформації
5. Виміри продуктивності коду – час виконання, використання пам’яті, …
6. Збір статистики з репозиторію – кількість комітів, кількість гілок, активність за певний період часу
7. Підтримка версіонування для коду, що зберігається не в репозиторіях source control
8. Можливість пошуку спеціальних зразків неправильного коду (налаштовуються користувачем)
9. Можливість створення скриптів для опису додаткових перевірок
10. Можливість створення плагінів для розширення функціональності

Навчальний проект

Навчальний проект дає можливість застосувати все вивчене в цьому (і попередніх) семестрах і створити більш-менш повноцінний та корисний застосунок.

Передбачається, що кожен студент вибере для себе цікаву тему проекту. Наведений список ідей – це лише підказки, можна обрати щось інше за погодженням з викладачем.

Можливі ідеї проектів

1. Розвиток однієї з лабораторних (більше функціональності, кращий інтерфейс користувача, об’єднання кількох лабораторних в один проект, …)
2. Математичні алгоритми
    a. Лінійна алгебра
    b. Вирішення рівнянь
    c. Диференційні рівняння
    d. Оптимізація функцій (пошук мінімумів/максимумів)
    e. Загальна алгебра, алгебраїчні структури
    f. Символьні обчислення, диференціювання, інтегрування, послідовності, ряди, …
    g. Теорія графів
    h. Теорія чисел
    i. Комбінаторика
    j. Теорія ймовірностей, математична статистика
3. Інструменти / утіліти
    a. Binary diff/patch
    b. Пошук копій файлів/каталогів

c. Робота з кількома репозиторіями
d. Віртуальна файлова система (каталоги всередині файлу, каталоги пошуку, різні точки зору/ієрархії каталогів, …)
e. Code review
f. Блокування якихось можливостей ОС, віруси/боротьба з ними
g. Використання можливостей ОС

4. Робота з текстом
    a. Аналіз тексту програми, пошук помилок, перевірка виконання рекомендацій/стилю
    b. Генерація програмного коду з певних шаблонів
    c. Видобування інформації з текстів природними мовами
    d. Пошук копій тексту/програмного коду, перевірка плагіату
5. Робота з зображеннями
    a. Розпізнавання образів, облич (face id)
    b. Виділення елементів зображення
    c. Перетворення зображень, фільтри
    d. Зняти на камеру кімнату, побудувати 3D модель
    e. Augmented reality (додавати елементи на зображення/відео з камери)
    f. Розпізнавання бланків форм
    g. Розпізнавання captcha
6. Робота з документами різних форматів
    a. Видобування інформації з документів (Word, Excel, PowerPoint, PDF, …)
    b. Генерація документів
    c. Перетворення документів з одного формату в інший
    d. Lite document viewer
7. Робота з мережею
    a. Завантаження та видобування інформації з веб сторінок
    b. Використання web API (наприклад, Google, Facebook, Twitter, …)
    c. Чат
    d. Передача файлів по мережі
    e. Мережева гра
    f. Brute force password break (multithreaded)
8. Паралельне програмування
    a. Мультипоточна реалізація алгоритмів чи структур даних
    b. Розподілені обчислення

c. GPGPU (обчислення на відеокартах), CUDA, …

9. Засоби для вивчення різних тем
    a. Система перевірки завдань з програмування
    b. Вивчення в ігровій формі
    c. Візуалізація алгоритмів/…
10. Моделювання наукових / технічних / соціальних / програмних систем
11. Візуалізація даних
    a. Графіки функцій
    b. Діаграми (2D, 3D)
12. Гра (з «штучним інтелектом»)
    a. Настільні ігри (шахи, шашки, …)
    b. Відео ігри
    c. Клітинкові автомати, Conway's Life

Додаткове завдання №1. Цікаві проблеми в коді

Описати якісь цікаві проблеми, баги, неочікувані ситуації, що виникають в коді. Це можуть бути типові проблеми, з якими студенти часто стикаються при написанні коду; або якась ситуація, з якою було важко розібратись; або демонстрація якоїсь незвичної поведінки, коли не одразу зрозуміло, як працює код, або ще щось подібне. Проблеми можуть стосуватись синтаксису та можливостей мови програмування, загальних принципів ООП, стандартних бібліотек, зовнішніх бібліотек чи фреймворків. Приклади бажано наводити для C++, але можна і для інших мов.

Опис можна наводити українською чи англійською мовою. Опис проблеми може бути в довільному форматі. Варто вказати наступну інформацію:

1. Короткий опис проблеми («назва»), за якою можна більш-менш зрозуміти, про що іде мова
2. Детальний опис, в чому саме полягає проблема чи незвична ситуація
3. Опис вирішення проблеми (якщо відомий). Можна наводити кілька варіантів вирішення проблеми, а також порівнювати їх – за це будуть виставлені ***додаткові бали***. Можна наводити навіть такі ситуації, для яких вирішення Вам невідомо.
4. Короткий приклад коду, що ілюструє дану проблему. Це має бути мінімальний приклад, з якого прибрано все зайве – але при цьому повний код, достатній для запуску (тобто мають бути всі #include, функція main() та/або юніт тести, що

запускають код і т.д.). Бажано наводити як варіанти коду з наявною проблемою, так і з вирішенням проблеми.

5. Посилання на репозиторії, де можна побачити цю проблему в реальному коді (якщо такі є). Це можуть бути завдання студентів (лабораторні, проекти) або якісь зовнішні приклади з відкритого коду. Необхідно вказати точне посилання – яка гілка, який коміт, який файл чи файли, які місця в файлах. Бажано вказати окремо коміти, в яких наявна ця проблема, та в яких проблема вирішена. Якщо немає комітів з невирішеною проблемою (наприклад, проблему вирішили без створення комітів з неправильним кодом) – варто вказати, як би виглядав код з цією проблемою. Аналогічно якщо немає комітів з вирішеною проблемою.
6. Посилання на документацію, статті, форуми, блог пости, StackOverflow і т.д., які якось описують цю або схожі проблеми, дають шляхи вирішення.
7. Ключові слова, за якими варто шукати інформацію про цю проблему.
8. Чому ця проблема здалась Вам цікавою?
9. Будь-які інші коментарі або додаткова інформація.

Не обов’язково наводити інформацію за усіма вказаними пунктами. Проте чим більше інформації буде наведено – тим більше балів буде виставлено за це завдання.

Кожен студент може описати кілька різних проблем. Чим більше різних описів буде надано – тим більше балів буде виставлено (звісно, з урахуванням якості наведених описів)

Лабораторна робота № 8.2. Індивідуальні та командні проекти

Мета роботи

Навчитися створювати цілісні проекти, використовуючи сучасні підходи до програмування. Для командних проектів – навчитися працювати в команді.

Загальні інструкції

1. Завдання необхідно виконувати в git репозиторії. Не треба створювати відкритий репозиторій – можна створити приватний репозиторій (на GitHub/Bitbucket/GitLab/…), і додати туди викладача, або створити локальний репозиторій і переслати його викладачу. За використання приватного репозиторію, а також за правильну роботу з ним (часті коміти, зрозумілі повідомлення в комітах, …) виставляються ***додаткові бали***.
2. Студенти можуть відслідковувати час виконання різних завдань – тобто десь зберігати інформацію, коли були виконані які завдання, скільки часу на це пішло і т.д. Це можна робити вручну (наприклад, зберігати в текстовому файлі чи таблиці записи про те коли, скільки часу та які завдання виконували). Або ж можна використовувати інструментальні засоби, наприклад зі списку https://en.wikipedia.org/wiki/Comparison_of_time-tracking_software або аналогічних. Бажано вказувати не лише яке із завдань 1-3 було виконано в певний проміжок часу, але також і над чим саме працювали (яку функціональність реалізовували, які недоліки виправляли, …). Студентам, які відслідковували час виконання завдань, будуть виставлені ***додаткові бали***. Інформація про витрати часу не буде впливати на оцінювання завдань контрольної роботи – оцінка буде залежати лише від якості реалізованого коду.
3. Кожне завдання варто робити у вигляді однієї програми (тобто один виконуваний файл, одна функція main(), довільна кількість файлів з кодом)
4. Демонстрацію роботи програми варто реалізувати таким чином, щоб все працювало без необхідності для користувача вводити якісь дані.
5. Якщо програма щось виводить – варто якось пояснити, що саме зараз виводиться.
6. Якщо демонстрацію роботи програми реалізовано у вигляді юніт тестів – за це будуть виставлені ***додаткові бали***.

7. В усіх завданнях (а особливо у завданні 1) треба намагатись реалізувати доцільний об’єктно-орієнтований дизайн, уникати дублювання коду. Також слід уникати типових недоліків, зокрема memory leaks.
8. Під час реалізації можна використовувати стандартні бібліотеки мови програмування, та/або якісь зовнішні бібліотеки, якщо це доцільно.
9. Завдання 2 розраховане на реалізацію з використанням шаблонів (C++ templates). Можна спробувати використати інші підходи, якщо це дозволить реалізувати потрібну функціональність. Якщо буде продемонстровано використання різних підходів, за це будуть виставлені ***додаткові бали***.
10. Щоб отримати максимальну кількість балів за завдання 3, треба не використовувати в своєму коді оператор delete.
11. У завданні 3, якщо сказано, що десь зберігаються екземпляри певного класу X – це можуть як екземпляри власне класу X (якщо він не є абстрактним), так і екземпляри його підкласів, доступ до яких виконується через інтерфейс класу X.
12. У завданні 3 використовується цілочисленне ділення.
13. Завдання 3 складається з 3 частин, за зростанням складності. Для отримання повних балів, треба реалізувати 1 та 2 частину; частина 3 дозволить отримати ***додаткові бали***.
14. Не варто копіювати код інших студентів – за це буде ставитись 0 балів. Краще спробувати щось написати самостійно – за це будуть хоча б якісь бали. Навіть якщо завдання виконане не повністю – якісь бали все одно будуть, тому краще зробити хоч щось, ніж не робити нічого чи намагатись списувати.

Варіанти завдань

Варіант 1

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов’язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія

повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості повідомлень.

Реалізувати алгоритм вибору повідомлень, які бачить заданий користувач. Повідомлення від друзів користувача впорядковуються за кількома критеріями: час публікації (новіші краще), рейтинг, відповідність темам користувача. З цих критеріїв розраховується зважений коефіцієнт для кожного повідомлення, і повідомлення впорядковуються за цим коефіцієнтом.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) = n! mod 101
2. Для від'ємних цілих чисел n, f(n) = n^3 mod 201
3. Для дійсних чисел d, f(d) = [exp(1/sin(d))] mod 301
4. Для текстових рядків s, f(s) = кількість голосних літер англійської мови в рядку s
5. Для пари p=(a,b), f(p) = $f(b)^{f(a)}$ mod 501
6. Для списку v=[v_1,…,v_k], f(v) = (f(v_1) f(v_2)+ f(v_2) f(v_3)+…+ f(v_{k-1}) f(v_k)) mod 701
7. Для всіх інших значень f(x) = 1751

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=S/2-3N+1, деструктор Alpha S:=S+N, деструктор Beta S:=S-2N, деструктор Gamma S:=S+N-1.

1. Реалізувати відповідні класи, створити і видалити певну кількість об'єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об'єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об'єктів.
3. Для заданої кількості об'єктів M (що включає як незалежні, так і вкладені об'єкти) перебрати всі можливі комбінації об'єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 2

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних завдань.

Реалізувати алгоритм створення рейтингового списку студентів. Для кожного предмета визначається список студентів, впорядкований за сумою отриманих балів за всі завдання цього предмета. Загальний рейтинговий список впорядковується за сумою позицій студента у списках з кожного предмета.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \sin(5n+102)$
2. Для від'ємних цілих чисел n, $f(n) = \log_2 |n^5 + 202|$
3. Для дійсних чисел d, $f(d) = d/(d+302)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(502f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \cos (v_i - 802)$
7. Для всіх інших значень f(x) = 35.02

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить

порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-2, деструктор Alpha S:=S-N, деструктор Beta S:=S+3N+2, деструктор Red S:=S+N/2, деструктор Green S:=S-N/2-1.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 3

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості перевезених вантажів.

Компанія-перевізник отримала можливість отримати деякі вулиці у пріоритетне користування. Компанія хоче обрати такі вулиці, щоб між будь-якими двома складами можна було доїхати лише по пріоритетних вулицях, і при цьому мінімізувати загальну довжину пріоритетних вулиць. Реалізувати алгоритм оптимального вибору таки пріоритетних вулиць.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n^2 \bmod 103$ англійською мовою
2. Для від'ємних цілих чисел n, $f(n) = (n^5+n^3) \bmod 203$ англійською мовою
3. Для дійсних чисел d, f(d) = sin(d+303) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = унікальні слова з усіх $f(v_i)$, у алфавітному порядку
7. Для всіх інших значень f(x) = "sorry please try again"

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може

повторюватись між різними базовими класами, починається з 1 для першого створеного екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N-3, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+3, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-2.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 4

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники не можуть переходити на інші проекти посередині роботи над проектом, і не можуть приєднуватись до проекту, якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних проектів.

Реалізувати алгоритм перевірки відсутності циклічних залежностей між проектами.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 104
2. Для від'ємних цілих чисел n, $f(n) = (n^4 + n^3) \bmod 204$ – цифри у представленні з основою 4
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 304$, k = 1,2,…, 42
4. Для текстових рядків s, $f(s) = k^k \bmod 404$, де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)
5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(b), повторений кількість разів відповідного елементу зі списку f(a)

6. Для списку v=[v_1,…,v_k], f(v) = об’єднання обернених списків f(v_i) ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [4,0,4,2]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+4,деструктор Alpha S:=S-N+4, деструктор Beta S:=S+N, деструктор Gamma S:=S+3N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 5

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості надісланих повідомлень.

Реалізувати алгоритм вибору найшвидшого шляху через сервери для доставки заданого повідомлення.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(2^n + n^2) \bmod 105$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 205$
3. Для дійсних чисел d, f(d) = синій, $[1/\sin(\log_2 d)] \bmod 305$
4. Для текстових рядків s, f(s) = зелений, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 3 до 5 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 505$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_k) + f(v_2)\, f(v_{k-1}) + \ldots + f(v_k)\, f(v_1)) \bmod 705$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) =червоний, 8395

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить

порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S+N-5, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+5, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-4.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 6

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості голосувань в парламенті.

Партії можуть об’єднуватись в блоки, якщо вони мають схожі позиції щодо законів. Задається певний рівень толерантності (ціле число), обчислюється кількість законів, які дві партії одночасно підтримують чи протистоять, від нього віднімається кількість законів, які одна з партій підтримує, а інша протистоїть, якщо сума перевищує рівень толерантності – партії можуть об’єднатись. Якщо партія А може об’єднатись з В, а В з С, то вони всі можуть об’єднатись, незалежно від близькості позицій А і С. Для заданого рівня толерантності, реалізувати алгоритм розрахунку кількості блоків, які можуть утворити всі партії, якщо вони максимально об’єднаються.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! + n - 1) \bmod 106$
2. Для від’ємних цілих чисел n, $f(n) = n^5 \bmod 206$

3. Для дійсних чисел d, f(d) = [1/sin(77d)] mod 306
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s
5. Для пари p=(a,b), f(p) = f(b)*(f(a)+1) mod 506
6. Для списку v=[v_1,…,v_k], f(v) = (f(v_1)+ f(v_2)+…+ f(v_k)) mod 706
7. Для всіх інших значень f(x) = 8497

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=3S+N+6, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+5, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-6.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 7

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, а також мають однакові типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості знайдених дефектів.

Реалізувати алгоритм, який для кожного компонента розраховує сумарну вартість всіх його підкомпонентів (включно з їх підкомпонентами, і т.д.)

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+107)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^5 + 207|$
3. Для дійсних чисел d, $f(d) = d/(d-307)$
4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(507f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \sin (v_i - 707)$

7. Для всіх інших значень f(x) = 12.07

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+7,деструктор Alpha S:=S-N+3, деструктор Beta S:=S+N, деструктор Gamma S:=S+2N-7.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 8

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов'язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості повідомлень.

Реалізувати алгоритм вибору повідомлень, які бачить заданий користувач. Повідомлення від друзів користувача впорядковуються за кількома критеріями: час публікації (новіші краще), рейтинг, відповідність темам користувача. За кожним критерієм будується впорядкований список повідомлень; задаються ймовірність вибору кожного зі списків; загальний список будується шляхом послідовного вибору елементів з кожного зі списків із заданою ймовірністю.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n! \bmod 108$
2. Для від'ємних цілих чисел n, $f(n) = n^7 \bmod 208$
3. Для дійсних чисел d, $f(d) = [\exp(1/\sin(d))] \bmod 308$
4. Для текстових рядків s, f(s) = кількість приголосних літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{f(a)} \bmod 508$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_2) + f(v_2)\, f(v_3) + \ldots + f(v_{k-1})\, f(v_k)) \bmod 708$
7. Для всіх інших значень f(x) = 4558

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для

кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=S/2-2N+8, деструктор Alpha S:=S+N, деструктор Beta S:=S-2N, деструктор Gamma S:=S-N-8.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 9

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних завдань.

Реалізувати алгоритм створення рейтингового списку студентів. Для кожного предмета визначається сума балів, а також ваговий коефіцієнт цього предмета. Загальний рейтинговий список впорядковується за сумою балів за предметами з відповідними ваговими коефіцієнтами.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \sin(5n+109)$
2. Для від'ємних цілих чисел n, $f(n) = \log_3 |n^3 + 209|$
3. Для дійсних чисел d, $f(d) = d/(d+309)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(509f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \cos (v_i - 809)$
7. Для всіх інших значень f(x) = 65.09

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить

порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-9, деструктор Alpha S:=S-N, деструктор Beta S:=S+3N+9, деструктор Red S:=S+N/2, деструктор Green S:=S-N/2-7.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 10

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості перевезених вантажів.

Реалізувати алгоритм пошуку оптимального шляху наземним транспортом між двома заданими складами.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n^2 \bmod 110$ англійською мовою
2. Для від'ємних цілих чисел n, $f(n) = (n^5+n^3) \bmod 210$ англійською мовою
3. Для дійсних чисел d, f(d) = sin(d+310) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = унікальні слова з усіх $f(v_i)$, у алфавітному порядку
7. Для всіх інших значень f(x) = "sorry please try again"

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може повторюватись між різними базовими класами, починається з 1 для першого створеного екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів

змінюють глобальну змінну S: деструктор Base1 S:=2S+N-10, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+10, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-9.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 11

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники не можуть переходити на інші проекти посередині роботи над проектом, але можуть приєднуватись до проекту, навіть якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних проектів.

В заданий момент часу, реалізувати алгоритм побудови списку ще не виконаних проектів, впорядкованих за наступними критеріями у порядку значимості: кількість технологій (спершу менше), складність (спершу простіші), сумарна ефективність всіх розробників, які можуть працювати над проектом (спершу більша) .

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 111
2. Для від'ємних цілих чисел n, $f(n) = (n^8 + n^3) \bmod 211$ – цифри у представленні з основою 4
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 311$, k = 1,2,…, 42
4. Для текстових рядків s, $f(s) = k^k \bmod 411$, де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)

5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(b), повторений кількість разів відповідного елементу зі списку f(a)
6. Для списку v=[v_1,…,v_k], f(v) = об’єднання обернених списків f(v_i) ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [4,0,71]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+11,деструктор Alpha S:=S-2N+11, деструктор Beta S:=S-N, деструктор Gamma S:=S+3N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 12

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості надісланих повідомлень.

Задається певний інтервал часу, і з системи виключаються всі сервери, які за цей час отримували повідомлення заданого типу. Реалізувати алгоритм перевірки, чи лишається можливість передачі повідомлення між довільною парою серверів, що не були виключені (повідомлення не можуть проходити через виключені сервери) .

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(2^n + n^2) \bmod 112$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 212$
3. Для дійсних чисел d, f(d) = синій, $[\ 1/\sin(\log_2 d)] \bmod 312$
4. Для текстових рядків s, f(s) = зелений, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 3 до 5 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 512$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_k) + f(v_2)\, f(v_{k-1}) + \ldots + f(v_k)\, f(v_1)) \bmod 712$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) =червоний, 8892

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-12, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+7, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-12.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 13

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості голосувань в парламенті.

Задається інтервал часу. Реалізувати алгоритм впорядкування партій за успішністю за цей проміжок часу – рейтинг успішності зростає для кожного закону за час, коли діють закони, які партія підтримує, і не діють закони, яким вона протистоїть, і навпаки, рейтинг падає, коли діють закони, яким партія протистоїть, і не діють закони, які партія підтримує.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! + n - 1) \bmod 113$
2. Для від’ємних цілих чисел n, $f(n) = n^5 \bmod 213$
3. Для дійсних чисел d, $f(d) = [1/\sin(77d)] \bmod 313$
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)*(f(a)+1) \bmod 513$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)+ f(v_2)+\ldots+ f(v_k)) \bmod 713$

7. Для всіх інших значень f(x) = 8403

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N+13, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+9, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-13.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 14

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, а також мають однакові типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості знайдених дефектів.

Задається тип компонентів. Реалізувати алгоритм, який для кожного компонента розраховує кількість підкомпонентів заданого типу (підкомпоненти довільного рівня вкладеності)

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+114)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^5 + 214|$
3. Для дійсних чисел d, $f(d) = d/(d-314)$
4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(514f(a))}$

6. Для списку v=[v_1,…,v_k], f(v) = max sin (v_i -714)
7. Для всіх інших значень f(x) = 10.14

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+14,деструктор Alpha S:=S-N+13, деструктор Beta S:=S+N, деструктор Gamma S:=S+2N-14.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 15

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов'язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості повідомлень.

Реалізувати алгоритм пошуку найкоротшого шляху через друзів між двома заданими користувачами.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n! \bmod 115$
2. Для від'ємних цілих чисел n, $f(n) = n^3 \bmod 215$
3. Для дійсних чисел d, $f(d) = [\exp(1/\sin(d))] \bmod 315$
4. Для текстових рядків s, f(s) = кількість голосних літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{f(a)} \bmod 515$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_2) + f(v_2)\, f(v_3) + \ldots + f(v_{k-1})\, f(v_k)) \bmod 715$
7. Для всіх інших значень f(x) = 7115

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=S/2-3N+15, деструктор Alpha S:=S+N, деструктор Beta S:=S-2N, деструктор Gamma S:=S+N-13.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 16

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних завдань.

Два студенти називаються «колегами», якщо в якийсь день вони одночасно працювали над певним завданням. Реалізувати алгоритм пошуку всіх студентів, які мають задану кількість колег відносно заданого завдання.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+16)$
2. Для від'ємних цілих чисел n, $f(n) = \log_2 |n^3 + 216|$
3. Для дійсних чисел d, $f(d) = d/(d+316)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(516f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \cos (v_i - 816)$
7. Для всіх інших значень f(x) = 83.16

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого

створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-16, деструктор Alpha S:=S-N, деструктор Beta S:=S+4N+16, деструктор Red S:=S+N/4, деструктор Green S:=S-N/2-8.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 17

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості перевезених вантажів.

Для заданого моменту часу, заданого вантажу, який знаходиться на заданому складі, реалізувати алгоритм створення списку транспортних засобів, впорядкованих відповідно до часу, потрібного їм для того, щоб забрати вантаж зі складу. Кожен транспортний засіб має завершити ту дію, яку він зараз виконує (пересування до наступного складу, завантаження чи розвантаження). Потім, якщо транспортний засіб не зможе взяти вантаж – він має продовжувати рух за своїм розкладом, при цьому не завантажуючи нові вантажі, поки не звільниться достатньо місця (за вагою та об'ємом). Нарешті, транспортний засіб має дістатись заданого складу. Якщо транспортний засіб взагалі не зможе перевозити вантаж (не вистачить ваги чи об'єму навіть без інших вантажів) – такі транспортні засоби додаються наприкінці списку.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) = n^2 mod 117 англійською мовою
2. Для від'ємних цілих чисел n, f(n) = (n^5+n^3) mod 217 англійською мовою
3. Для дійсних чисел d, f(d) = sin(d+317) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)

6. Для списку v=[v_1,…,v_k], f(v) = унікальні слова з усіх f(v_i), у алфавітному порядку
7. Для всіх інших значень f(x) = “sorry try again please”

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може повторюватись між різними базовими класами, починається з 1 для першого створеного екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N-17, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+17, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-7.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 18

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники можуть переходити на інші проекти посередині роботи над проектом (якщо проект, з якого вони переходять, може виконуватись і без них), але не можуть приєднуватись до проекту, якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних проектів.

Розробник A може тренувати розробника B (який в цьому випадку називається учнем A), якщо вони знають хоча б одну спільну технологію, і при цьому ефективність A більша за B. Реалізувати алгоритм, який починаючи із заданого розробника, виводить всіх учнів цього розробника, потім всіх учнів цих учнів і т.д.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 118
2. Для від'ємних цілих чисел n, $f(n) = (n^4 + n^3) \bmod 218$ – цифри у представленні з основою 4
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 318$, k = 1,2,…, 42

4. Для текстових рядків s, $f(s) = k^k \bmod 418$, де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)
5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(b), повторений кількість разів відповідного елементу зі списку f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = об'єднання обернених списків $f(v_i)$ ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [4,0,7, 0, 8]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+18,деструктор Alpha S:=S-N+18, деструктор Beta S:=S+N, деструктор Gamma S:=S+3N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об'єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об'єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об'єктів.
3. Для заданої кількості об'єктів M (що включає як незалежні, так і вкладені об'єкти) перебрати всі можливі комбінації об'єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 19

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості надісланих повідомлень.

Для заданого проміжку часу та списку типів повідомлень, реалізувати алгоритм впорядкування серверів за кількістю повідомлень одного з цих типів, які сервер отримав чи надіслав за цей проміжок часу.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(2^n + n^2) \bmod 119$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 219$
3. Для дійсних чисел d, f(d) = синій, $[\ 1/\sin(\log_2 d)] \bmod 319$
4. Для текстових рядків s, f(s) = зелений, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 4 до 7 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 519$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\ f(v_k) + f(v_2)\ f(v_{k-1}) + \ldots + f(v_k)\ f(v_1)) \bmod 719$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) =червоний, 8359

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S+N-19, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+5, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-4.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 20

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості голосувань в парламенті.

Партії можуть об’єднуватись в блоки, якщо вони мають схожі позиції щодо законів. Задається певний рівень толерантності (ціле число), обчислюється кількість законів, які дві партії одночасно підтримують чи протистоять, від нього віднімається кількість законів, які одна з партій підтримує, а інша протистоїть, якщо сума перевищує рівень толерантності – партії можуть об’єднатись. Для заданого рівня толерантності і двох заданих партій, реалізувати алгоритм пошуку мінімальної кількості проміжних партій, так що задані партії можуть об’єднатись через задані проміжні партії (алгоритм має повідомити, якщо для заданого рівня толерантності об’єднання взагалі неможливе)

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! + n - 1) \bmod 120$
2. Для від’ємних цілих чисел n, $f(n) = n^5 \bmod 220$

3. Для дійсних чисел d, f(d) = [1/sin(87d)] mod 320
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s
5. Для пари p=(a,b), f(p) = f(b)*(f(a)+1) mod 520
6. Для списку v=[v_1,…,v_k], f(v) = (f(v_1)+ f(v_2)+…+ f(v_k)) mod 720
7. Для всіх інших значень f(x) = 8002

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=3S+N+20, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+5, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-20.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 21

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, але можуть мати різні типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості знайдених дефектів.

Реалізувати алгоритм, який для заданого типу дефекту розраховує збитки у винагороді залежно від того, в якому компоненті станеться цей дефект (розглядаються лише компоненти правильного типу, збитки за один період часу без перевірки/виправлення/заміни). Алгоритм має повертати список компонентів, впорядкований за збитками (від більших до менших).

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+121)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^5 + 221|$
3. Для дійсних чисел d, $f(d) = d/(d+321)$

4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(521f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = max sin (v_i -721)
7. Для всіх інших значень f(x) = 12.21

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+21,деструктор Alpha S:=S-N+5, деструктор Beta S:=S+N, деструктор Gamma S:=S+2N-21.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 22

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов'язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості повідомлень.

Для заданої теми, реалізувати алгоритм впорядкування користувачів відповідно до активності в цій темі. Користувачі отримують фіксовану кількість балів за різні види активності, пов'язаної з даною темою – створення нових повідомлень, відповідь на повідомлення, пересилання повідомлення, зміна рейтингу повідомлень. Список будується за загальною кількістю балів, від більшої до меншої.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n! \bmod 122$
2. Для від'ємних цілих чисел n, $f(n) = n^3 \bmod 222$
3. Для дійсних чисел d, $f(d) = [\exp(1/\sin(d))] \bmod 322$
4. Для текстових рядків s, f(s) = кількість голосних літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{f(a)} \bmod 522$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_2) + f(v_2)\, f(v_3) + \ldots + f(v_{k-1})\, f(v_k)) \bmod 722$
7. Для всіх інших значень f(x) = 1782

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори

класів змінюють глобальну змінну S: деструктор Base S:=S/2-2N+22, деструктор Alpha S:=S+3N, деструктор Beta S:=S-2N, деструктор Gamma S:=S+N-22.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 23

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних завдань.

Задається період часу та список знань. Реалізувати алгоритм створення списку студентів, які за цей період підвищили рівень цих знань, впорядкований за сумарним зростанням рівня знань (від більшого до меншого).

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \sin(5n+123)$
2. Для від'ємних цілих чисел n, $f(n) = \log_2 |n^5 + 223|$
3. Для дійсних чисел d, $f(d) = d/(d+323)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(523f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \cos (v_i - 823)$
7. Для всіх інших значень f(x) = 35.23

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого

створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-23, деструктор Alpha S:=S-N, деструктор Beta S:=S+3N+23, деструктор Red S:=S+N/2, деструктор Green S:=S-N/2-11.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 24

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості перевезених вантажів.

Наземним транспортним засобам забороняється відвідувати склади, з яких за заданий період часу відправляли вантажі заданого типу (не можна навіть проїздити повз такі склади, не зупиняючись на них). Транспортні засоби, які на закінчення цього періоду часу знаходились на одному з цих складів, або їхали з них чи до них, або везли вантаж цього типу, вилучаються з системи. Реалізувати алгоритм знаходження груп складів, між якими тепер неможливо проїхати наземним транспортом, а також кількість транспортних засобів (окремо наземних та повітряних), які залишились в кожній групі складів.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n^2 \bmod 124$ англійською мовою
2. Для від'ємних цілих чисел n, $f(n) = (n^7+n^3) \bmod 224$ англійською мовою
3. Для дійсних чисел d, f(d) = sin(d+324) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = унікальні слова з усіх $f(v_i)$, у алфавітному порядку
7. Для всіх інших значень f(x) = "no luck this time"

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може повторюватись між різними базовими класами, починається з 1 для першого створеного екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N-24, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+24, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-9.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 25

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники не можуть переходити на інші проекти посередині роботи над проектом, і не можуть приєднуватись до проекту, якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних проектів.

Реалізувати алгоритм побудови списку технологій, впорядкованих за наступними критеріями: 1) кількість пов’язаних проектів; 2) кількість розробників, які знають цю технологію; 3) загальний час виконання проектів, пов’язаних з технологією. Будується спільне значення, що об’єднує ці критерії з певними ваговими коефіцієнтами, і технології впорядковуються за цим значенням.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 125
2. Для від’ємних цілих чисел n, $f(n) = (n^4 + n^3) \bmod 225$ – цифри у представленні з основою 5
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 325$, k = 1,2,…, 42

4. Для текстових рядків s, $f(s) = k^k \bmod 425$, де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)
5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(b), повторений кількість разів відповідного елементу зі списку f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = об'єднання обернених списків $f(v_i)$ ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [4,0,25]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+25,деструктор Alpha S:=S-N, деструктор Beta S:=S+N+25, деструктор Gamma S:=S+3N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об'єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об'єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об'єктів.
3. Для заданої кількості об'єктів M (що включає як незалежні, так і вкладені об'єкти) перебрати всі можливі комбінації об'єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 26

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості надісланих повідомлень.

Адміністратори мережі мають можливість прокласти більш сучасні надшвидкі зв'язки між окремими серверами (лише там, де вже існували зв'язки раніше). Адміністратори хочуть обрати такі зв'язки, щоб 1) загальна кількість нових зв'язків була мінімальною, але при цьому між будь-якими серверами можна було доставити повідомлення лише користуючись новими зв'язками; 2) сумарна кількість повідомлень, яка проходила раніше по цим зв'язкам, була максимальною. Реалізувати алгоритм вибору зв'язків для покращення відповідно до цих умов.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(2^n + n^2) \bmod 126$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 226$
3. Для дійсних чисел d, f(d) = синій, $[1/\sin(\log_2 d)] \bmod 326$
4. Для текстових рядків s, f(s) = зелений, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 2 до 6 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 526$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох

6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_k) + f(v_2)\, f(v_{k-1}) + \ldots + f(v_k)\, f(v_1)) \bmod 726$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) =червоний, 9386

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S+N-26, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+26, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-14.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати $M<7$.

Варіант 27

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості голосувань в парламенті.

Реалізувати алгоритм побудови списку законів, впорядкованих за наступними критеріями (у всіх критеріях спершу більше): 1) кількість виборців, які їх підтримують; 2) кількість партій, які їх підтримують; 3) кількість виборців, які їм протистоять; 4) кількість партій, які їм протистоять; 5) кількість розглядів закону в парламенті. Будується спільне значення, що об’єднує ці критерії з певними ваговими коефіцієнтами, і закони впорядковуються за цим значенням.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) = (n! + n -1) mod 127
2. Для від’ємних цілих чисел n, f(n) = n^5 mod 227
3. Для дійсних чисел d, f(d) = [1/sin(77d)] mod 327
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s

5. Для пари p=(a,b), f(p) = f(b)*(f(a)+3) mod 527
6. Для списку v=[v_1,…,v_k], f(v) = (f(v_1)+ f(v_2)+…+ f(v_k)) mod 727
7. Для всіх інших значень f(x) = 7487

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=3S+N+27, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+5, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-27.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 28

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, а також мають однакові типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості знайдених дефектів.

Реалізувати алгоритм, який заданої функції повертає список всіх компонентів, які її виконують. Спершу в списку компоненти найвищого рівня, потім їх підкомпоненти, потім підкомпоненти цих підкомпонентів і т.д.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(7n+128)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^5 + 228|$
3. Для дійсних чисел d, $f(d) = d/(d-528)$
4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(528f(a))}$

6. Для списку v=[v_1,…,v_k], f(v) = max sin (v_i -728)
7. Для всіх інших значень f(x) = 12.28

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+28,деструктор Alpha S:=S-N+13, деструктор Beta S:=S+N-28, деструктор Gamma S:=S+2N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 29

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов'язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості повідомлень.

Для заданого користувача задана тема стає «забороненою» – цей користувач перестає спілкуватись з тими, хто пов'язаний з цією темою (тобто має її серед улюблених, або створював повідомлення на цю тему). Реалізувати алгоритм, який будує список користувачів, з якими даний користувач ще може користуватись – спершу друзі, потім друзі друзів і т.д., наприкінці списку всі користувачі, хто не пов'язані з даним користувачем через друзів (але все одно можуть спілкуватись, тобто не пов'язані із забороненою темою).

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n! \bmod 129$
2. Для від'ємних цілих чисел n, $f(n) = n^3 \bmod 229$
3. Для дійсних чисел d, $f(d) = [\exp(1/\sin(d))] \bmod 329$
4. Для текстових рядків s, f(s) = кількість голосних літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{f(a)} \bmod 529$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_2) + f(v_2)\, f(v_3) + \ldots + f(v_{k-1})\, f(v_k)) \bmod 729$
7. Для всіх інших значень f(x) = 1799

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для

кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=S/2-3N+29, деструктор Alpha S:=S+N, деструктор Beta S:=S-2N-17, деструктор Gamma S:=S+N-29.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 30

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних завдань.

Для двох заданих знань, їх близькість – це кількість завдань, які одночасно пов'язані з обома знаннями. Реалізувати алгоритм побудови дерева знань, так щоб сумарна близькість між сусідніми знаннями дерева була максимальною.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \sin(5n+130)$
2. Для від'ємних цілих чисел n, $f(n) = \log_2 |n^5 + 230|$
3. Для дійсних чисел d, $f(d) = d/(d+330)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(530f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \min \cos (v_i - 830)$
7. Для всіх інших значень f(x) = 35.03

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого

створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-30, деструктор Alpha S:=S-N, деструктор Beta S:=S+3N+30, деструктор Red S:=S+N/2, деструктор Green S:=S-N/2+14.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 31

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості перевезених вантажів.

Реалізувати алгоритм побудови списку транспортних засобів (всіх типів), які доставили найбільшу кількість вантажів заданого типу за заданий проміжок часу. Порядок за кількістю вантажів (від більшої до меншої); для однакової кількості за загальною вагою вантажів цього типу; для однакової кількості та ваги за загальним об'ємом.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n^2 \bmod 131$ англійською мовою
2. Для від'ємних цілих чисел n, $f(n) = (n^5+n^3) \bmod 231$ англійською мовою
3. Для дійсних чисел d, f(d) = sin(d+331) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = унікальні слова з усіх $f(v_i)$, у алфавітному порядку
7. Для всіх інших значень f(x) = "sorry maybe try again"

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може повторюватись між різними базовими класами, починається з 1 для першого створеного

екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N-31, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+31, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-11.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 32

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники не можуть переходити на інші проекти посередині роботи над проектом, але можуть приєднуватись до проекту, навіть якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних проектів.

Реалізувати алгоритм побудови списку технологій, впорядкованих за наступними критеріями: 1) кількість пов’язаних проектів; 2) кількість розробників, які знають цю технологію; 3) загальний час виконання проектів, пов’язаних з технологією. Для кожної технології визначається позиція в кожному з цих списків, і позиція у загальному списку визначається сумою позицій в кожному списку.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 132
2. Для від’ємних цілих чисел n, $f(n) = (n^4 + n^3) \bmod 232$ – цифри у представленні з основою 7
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 332$, k = 1,2,…, 42

4. Для текстових рядків s, $f(s) = k^k \bmod 432$, де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)
5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(a), повторений кількість разів відповідного елементу зі списку f(b)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = об'єднання обернених списків $f(v_i)$ ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [4,0,23]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+32,деструктор Alpha S:=S-N+4, деструктор Beta S:=S+N, деструктор Gamma S:=S+3N-32.

1. Реалізувати відповідні класи, створити і видалити певну кількість об'єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об'єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об'єктів.
3. Для заданої кількості об'єктів M (що включає як незалежні, так і вкладені об'єкти) перебрати всі можливі комбінації об'єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 33

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості надісланих повідомлень.

Для заданої програми та заданого проміжку часу, реалізувати алгоритм побудови списку серверів, через які найчастіше проходили повідомлення від програми та до неї. Список впорядкований за кількістю повідомлень та загальним розміром повідомлень.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(2^n + n^2) \bmod 133$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 233$
3. Для дійсних чисел d, f(d) = синій, $[\ 1/\sin(\log_2 d)] \bmod 333$
4. Для текстових рядків s, f(s) = зелений, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 3 до 5 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 533$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\ f(v_k) + f(v_2)\ f(v_{k-1}) + \ldots + f(v_k)\ f(v_1)) \bmod 733$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) =червоний, 9583

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S+N-33, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+33, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-24.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 34

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості голосувань в парламенті.

Реалізувати алгоритм побудови списку законів, впорядкованих за наступними критеріями (у всіх критеріях спершу більше): 1) кількість виборців, які їх підтримують; 2) кількість партій, які їх підтримують; 3) кількість виборців, які їм протистоять; 4) кількість партій, які їм протистоять; 5) кількість розглядів закону в парламенті. Для кожного закону визначається позиція в кожному з цих списків, і позиція у загальному списку визначається сумою позицій в кожному списку.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) = (n! + n -1) mod 134
2. Для від’ємних цілих чисел n, f(n) = n^5 mod 234
3. Для дійсних чисел d, f(d) = [1/sin(77d)] mod 334
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s

5. Для пари p=(a,b), f(p) = f(b)*(f(a)+1) mod 534
6. Для списку v=[v_1,…,v_k], f(v) = (f(v_1)+ f(v_2)+…+ f(v_k)) mod 734
7. Для всіх інших значень f(x) = 8974

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=3S+N+34, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+34, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-13.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 35

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, а також мають однакові типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості знайдених дефектів.

Реалізувати алгоритм, який будує список дефектів, впорядкованих за часом існування (від виникнення до усунення шляхом виправлення чи заміни компоненту). Порядок від більшого до меншого часу. Дефекти, які на момент створення списку виявлені, але не виправлені, знаходяться на початку списку. Дефекти, які на момент створення списку виникли, але не були виявлені, до списку не потрапляють.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+135)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^5 + 235|$
3. Для дійсних чисел d, $f(d) = d/(d-335)$

4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(535f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = max sin (v_i -735)
7. Для всіх інших значень f(x) = 12.35

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+35,деструктор Alpha S:=S-N+35, деструктор Beta S:=S+N, деструктор Gamma S:=S+2N-17.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 36

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов'язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості повідомлень.

Для заданого повідомлення, реалізувати алгоритм пошуку користувачів, які ще не взаємодіяли з ним (тобто не автор, немає посилання на них, не відповідали, не пересилали повідомлення і не оцінювали). Починаючи з користувачів, які взаємодіяли з повідомленням, будуються списки їх друзів, потім друзів друзів і т.д. Порядок користувачів залежить в першу чергу від ступені участі тих, хто взаємодіяли, і вже потім від рівнів дружби – тобто друзі друзів автора (і їх друзі, і т.д.) у списку раніше, ніж друзі тих, хто лише оцінювали повідомлення.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n! \bmod 136$
2. Для від'ємних цілих чисел n, $f(n) = n^3 \bmod 236$
3. Для дійсних чисел d, $f(d) = [\exp(1/\sin(d))] \bmod 336$
4. Для текстових рядків s, f(s) = кількість голосних літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{f(a)} \bmod 536$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_2) + f(v_2)\, f(v_3) + \ldots + f(v_{k-1})\, f(v_k)) \bmod 736$
7. Для всіх інших значень f(x) = 1786

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний

поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=S/2-3N+36, деструктор Alpha S:=S+N, деструктор Beta S:=S-2N, деструктор Gamma S:=S+N-36.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 37

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних завдань.

Реалізувати алгоритм побудови списку найскладніших завдань. Задаються кілька критеріїв: 1) сума часу, яке витратили всі студенти на виконання цього завдання, 2) загальне зростання знань всіх студентів в результаті виконання цього завдання, 3) рівень складності завдання. Складаються впорядковані списки по кожному з критеріїв. Загальний список будується наступним чином: беруть перший елемент першого списку, потім перший елемент другого, потім перший елемент третього, потім другий елемент першого списку і т.д. (уникаючи повторів).

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \sin(5n+137)$
2. Для від'ємних цілих чисел n, $f(n) = \log_2 (n^5 + 237)$
3. Для дійсних чисел d, $f(d) = d/(d+337)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(537f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \cos (v_i - 837)$

7. Для всіх інших значень f(x) = 35.47

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-37, деструктор Alpha S:=S-N, деструктор Beta S:=S+3N+37, деструктор Red S:=S+N/2, деструктор Green S:=S-N/2-17.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 38

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості перевезених вантажів.

Для заданого транспортного засобу і заданого періоду часу, реалізувати алгоритм побудови списків типів вантажу, який даний засіб доставляв найбільше (впорядкований за кількістю та вагою вантажів), та складів, який він відвідував найчастіше (впорядкований за кількістю відвідувань та сумарною вагою вантажів, доставлених до цього складу та з цього складу).

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n^2 \bmod 138$ англійською мовою
2. Для від'ємних цілих чисел n, $f(n) = (n^5+n^3) \bmod 238$ англійською мовою
3. Для дійсних чисел d, f(d) = sin(d+338) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = унікальні слова з усіх $f(v_i)$, у алфавітному порядку
7. Для всіх інших значень f(x) = "sorry please try again"

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може

повторюватись між різними базовими класами, починається з 1 для першого створеного екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N-38, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+38, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-21.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 39

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники можуть переходити на інші проекти посередині роботи над проектом (якщо проект, з якого вони переходять, може виконуватись і без них), але не можуть приєднуватись до проекту, якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості виконаних проектів.

Для двох заданих технологій, їх близькість – це кількість проектів, які одночасно їх використовують плюс кількість розробників, які одночасно ними володіють. Реалізувати алгоритм побудови дерева технологій, так щоб сумарна близькість між сусідніми технологіями дерева була максимальною.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 139
2. Для від'ємних цілих чисел n, $f(n) = (n^4 + n^3) \bmod 239$ – цифри у представленні з основою 4
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 339$, k = 1,2,…, 42

4. Для текстових рядків s, f(s) = k^k mod 439 , де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)
5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(b), повторений кількість разів відповідного елементу зі списку f(a)
6. Для списку v=[v_1,…,v_k], f(v) = об’єднання обернених списків f(v_i) ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [4,0,4,9]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+39,деструктор Alpha S:=S-N+39, деструктор Beta S:=S+N, деструктор Gamma S:=S+3N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 40

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості надісланих повідомлень.

Для заданого списку типів повідомлень, реалізувати алгоритм побудови 1) списку серверів, через які найчастіше пересилались повідомлення цих типів, та 2) списку програм, які найчастіше відправляли чи отримували повідомлення цих типів. Списки впорядковані за кількістю повідомлень та сумарним розміром повідомлень.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(2^n + n^2) \bmod 140$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 240$
3. Для дійсних чисел d, f(d) = синій, $[\ 1/\sin(\log_2 d)] \bmod 340$
4. Для текстових рядків s, f(s) = зелений, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 2 до 4 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 540$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\ f(v_k) + f(v_2)\ f(v_{k-1}) + \ldots + f(v_k)\ f(v_1)) \bmod 740$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) = синій, 3890

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=4S+N-40, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+23, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-40.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 41

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості голосувань в парламенті.

Задається інтервал часу. Реалізувати алгоритм впорядкування виборців за рівнем задоволення діями партій за цей проміжок часу. Рівень задоволення зростає для кожного закону за час, коли діють закони, які підтримує цей виборець і партія, за яку він голосував, і не діють закони, яким вони протистоять, і навпаки, рівень задоволення падає, коли діють закони, яким виборець та його партія протистоять, і не діють закони, які вони підтримують.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) = (n! + n -1) mod 141
2. Для від’ємних цілих чисел n, f(n) = n^3 mod 241
3. Для дійсних чисел d, f(d) = [1/sin(77d)] mod 341
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s
5. Для пари p=(a,b), f(p) = f(b)*(f(a)+1) mod 541

6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)+ f(v_2)+\ldots+ f(v_k)) \bmod 741$
7. Для всіх інших значень $f(x) = 8941$

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=3S+N+41, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+14, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-41.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати $M<5$.

Варіант 42

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, а також мають однакові типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості знайдених дефектів.

Реалізувати алгоритм, який для даної функції виводить список компонентів, які її виконують. Список починається з компонентів найвищого рівня, потім їх підкомпоненти, потім підкомпоненти цих підкомпонентів і т.д.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+142)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^5 + 242|$
3. Для дійсних чисел d, $f(d) = d/(d-342)$
4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(542f(a))}$

6. Для списку v=[v_1,…,v_k], f(v) = max sin (v_i -742)
7. Для всіх інших значень f(x) = 42.12

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+42,деструктор Alpha S:=S-N+3, деструктор Beta S:=S+N, деструктор Gamma S:=S+3N-42.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 43

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов'язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості оцінювань.

Реалізувати алгоритм вибору повідомлень, які бачить заданий користувач. Повідомлення від друзів користувача впорядковуються за кількома критеріями: час публікації (новіші краще), рейтинг, відповідність темам користувача. З цих критеріїв розраховується зважений коефіцієнт для кожного повідомлення, і повідомлення впорядковуються за цим коефіцієнтом.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n! - 1 \bmod 143$
2. Для від'ємних цілих чисел n, $f(n) = n^3 \bmod 243$
3. Для дійсних чисел d, $f(d) = [\exp(1/\sin(d))] \bmod 343$
4. Для текстових рядків s, f(s) = кількість приголосних літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{f(a)} \bmod 543$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_2) + f(v_2)\, f(v_3) + \ldots + f(v_{k-1})\, f(v_k)) \bmod 743$
7. Для всіх інших значень f(x) = 1251

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори

класів змінюють глобальну змінну S: деструктор Base S:=S/2-3N-3, деструктор Alpha S:=S+N, деструктор Beta S:=S-2N, деструктор Gamma S:=S+N-1.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 44

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості балів (набраних всіма студентами за усі виконані завдання).

Реалізувати алгоритм створення рейтингового списку студентів. Для кожного предмета визначається список студентів, впорядкований за сумою отриманих балів за всі завдання цього предмета. Загальний рейтинговий список впорядковується за сумою позицій студента у списках з кожного предмета.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \sin(5n-144)$
2. Для від'ємних цілих чисел n, $f(n) = \log_2 |n^5 + 244|$
3. Для дійсних чисел d, $f(d) = d/(d+344)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(1-544f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \cos (v_i - 844)$
7. Для всіх інших значень f(x) = 35.44

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить

порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-2, деструктор Alpha S:=S-N, деструктор Beta S:=S+3N+2, деструктор Red S:=S+N/2, деструктор Green S:=S-N/2-1.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 45

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної ваги перевезених вантажів.

Компанія-перевізник отримала можливість отримати деякі вулиці у пріоритетне користування. Компанія хоче обрати такі вулиці, щоб між будь-якими двома складами можна було доїхати лише по пріоритетних вулицях, і при цьому мінімізувати загальну довжину пріоритетних вулиць. Реалізувати алгоритм оптимального вибору таки пріоритетних вулиць.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n^2 \bmod 145$ англійською мовою
2. Для від'ємних цілих чисел n, $f(n) = (n^5-n^3) \bmod 245$ англійською мовою
3. Для дійсних чисел d, f(d) = sin(d-345) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = унікальні слова з усіх $f(v_i)$, у алфавітному порядку
7. Для всіх інших значень f(x) = "sorry please try again"

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може

повторюватись між різними базовими класами, починається з 1 для першого створеного екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N-3, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+3, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-2.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 46

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники не можуть переходити на інші проекти посередині роботи над проектом, і не можуть приєднуватись до проекту, якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості “людино-днів”, витрачених на проекти.

Реалізувати алгоритм перевірки відсутності циклічних залежностей між проектами.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 146
2. Для від’ємних цілих чисел n, $f(n) = (n^4 + n^3) \bmod 246$ – цифри у представленні з основою 4
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 346$, k = 1,2,…, 37
4. Для текстових рядків s, $f(s) = k^k \bmod 446$, де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)
5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(b), повторений кількість разів відповідного елементу зі списку f(a)

6. Для списку v=[v_1,…,v_k], f(v) = об’єднання обернених списків f(v_i) ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [4,0,4,2]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+4,деструктор Alpha S:=S-N+4, деструктор Beta S:=S+N, деструктор Gamma S:=S+3N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 47

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального розміру повністю отриманих повідомлень.

Реалізувати алгоритм вибору найшвидшого шляху через сервери для доставки заданого повідомлення.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(2^n + n^2) \bmod 147$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 247$
3. Для дійсних чисел d, f(d) = синій, $[1/\sin(\log_2 d)] \bmod 347$
4. Для текстових рядків s, f(s) = зелений, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 3 до 5 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 547$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_k) + f(v_2)\, f(v_{k-1}) + \ldots + f(v_k)\, f(v_1)) \bmod 747$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) =червоний, 8387

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить

порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S+N-5, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+5, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-4.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 48

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального часу, витраченого на розгляд законів.

Партії можуть об’єднуватись в блоки, якщо вони мають схожі позиції щодо законів. Задається певний рівень толерантності (ціле число), обчислюється кількість законів, які дві партії одночасно підтримують чи протистоять, від нього віднімається кількість законів, які одна з партій підтримує, а інша протистоїть, якщо сума перевищує рівень толерантності – партії можуть об’єднатись. Якщо партія А може об’єднатись з В, а В з С, то вони всі можуть об’єднатись, незалежно від близькості позицій А і С. Для заданого рівня толерантності, реалізувати алгоритм розрахунку кількості блоків, які можуть утворити всі партії, якщо вони максимально об’єднаються.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) = (n! - n -1) mod 148
2. Для від’ємних цілих чисел n, f(n) = n^5 mod 248

3. Для дійсних чисел d, f(d) = [1/sin(77d)] mod 348
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s
5. Для пари p=(a,b), f(p) = f(b)*(f(a)+1) mod 548
6. Для списку v=[v_1,…,v_k], f(v) = (f(v_1)+ f(v_2)+…+ f(v_k)) mod 748
7. Для всіх інших значень f(x) = 8497

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=3S+N+6, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+5, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-6.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 49

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, а також мають однакові типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної вартості, витраченої на виправлення дефектів.

Реалізувати алгоритм, який для кожного компонента розраховує сумарну вартість всіх його підкомпонентів (включно з їх підкомпонентами, і т.д.)

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+149)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^5 + 249|$
3. Для дійсних чисел d, $f(d) = d/(d-349)$
4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(549f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \sin (v_i - 749)$

7. Для всіх інших значень f(x) = 15.49

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+7,деструктор Alpha S:=S-N+3, деструктор Beta S:=S+N, деструктор Gamma S:=S+2N-7.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 50

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов'язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості оцінювань.

Реалізувати алгоритм вибору повідомлень, які бачить заданий користувач. Повідомлення від друзів користувача впорядковуються за кількома критеріями: час публікації (новіші краще), рейтинг, відповідність темам користувача. За кожним критерієм будується впорядкований список повідомлень; задаються ймовірність вибору кожного зі списків; загальний список будується шляхом послідовного вибору елементів з кожного зі списків із заданою ймовірністю.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n! \bmod 150$
2. Для від'ємних цілих чисел n, $f(n) = n^7 \bmod 250$
3. Для дійсних чисел d, $f(d) = [\exp(1/\sin(d))] \bmod 350$
4. Для текстових рядків s, f(s) = кількість приголосних літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{f(a)} \bmod 550$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_2) + f(v_2)\, f(v_3) + \ldots + f(v_{k-1})\, f(v_k)) \bmod 850$
7. Для всіх інших значень f(x) = 4585

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для

кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=S/2-2N+8, деструктор Alpha S:=S+N, деструктор Beta S:=S-2N, деструктор Gamma S:=S-N-8.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 51

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості балів (набраних всіма студентами за усі виконані завдання).

Реалізувати алгоритм створення рейтингового списку студентів. Для кожного предмета визначається сума балів, а також ваговий коефіцієнт цього предмета. Загальний рейтинговий список впорядковується за сумою балів за предметами з відповідними ваговими коефіцієнтами.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \sin(5n+151)$
2. Для від'ємних цілих чисел n, $f(n) = \log_3 |n^3 + 251|$
3. Для дійсних чисел d, $f(d) = d/(d+351)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(551f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \cos (v_i - 851)$
7. Для всіх інших значень f(x) = 69.51

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить

порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-9, деструктор Alpha S:=S-N, деструктор Beta S:=S+3N+9, деструктор Red S:=S+N/2, деструктор Green S:=S-N/2-7.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 52

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної ваги перевезених вантажів.

Реалізувати алгоритм пошуку оптимального шляху наземним транспортом між двома заданими складами.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n^2 \bmod 152$ англійською мовою
2. Для від'ємних цілих чисел n, $f(n) = (n^5+n^3) \bmod 252$ англійською мовою
3. Для дійсних чисел d, f(d) = sin(d+352) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = унікальні слова з усіх $f(v_i)$, у алфавітному порядку
7. Для всіх інших значень f(x) = "sorry please try again"

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може повторюватись між різними базовими класами, починається з 1 для першого створеного екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів

змінюють глобальну змінну S: деструктор Base1 S:=2S+N-10, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+10, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-9.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 53

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники не можуть переходити на інші проекти посередині роботи над проектом, але можуть приєднуватись до проекту, навіть якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості “людино-днів”, витрачених на проекти.

В заданий момент часу, реалізувати алгоритм побудови списку ще не виконаних проектів, впорядкованих за наступними критеріями у порядку значимості: кількість технологій (спершу менше), складність (спершу простіші), сумарна ефективність всіх розробників, які можуть працювати над проектом (спершу більша) .

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 153
2. Для від’ємних цілих чисел n, $f(n) = (n^8 + n^3) \bmod 253$ – цифри у представленні з основою 4
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 353$, k = 1,2,…, 29
4. Для текстових рядків s, $f(s) = k^k \bmod 453$, де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)

5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(b), повторений кількість разів відповідного елементу зі списку f(a)
6. Для списку v=[v_1,…,v_k], f(v) = об’єднання обернених списків f(v_i) ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [42,0,35]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+11,деструктор Alpha S:=S-2N+11, деструктор Beta S:=S-N, деструктор Gamma S:=S+3N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 54

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального розміру повністю отриманих повідомлень.

Задається певний інтервал часу, і з системи виключаються всі сервери, які за цей час отримували повідомлення заданого типу. Реалізувати алгоритм перевірки, чи лишається можливість передачі повідомлення між довільною парою серверів, що не були виключені (повідомлення не можуть проходити через виключені сервери) .

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(3^n - n^3) \bmod 154$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 254$
3. Для дійсних чисел d, f(d) = синій, $[\ 1/\sin(\log_2 d)] \bmod 354$
4. Для текстових рядків s, f(s) = зелений, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 3 до 5 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 554$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\ f(v_k) + f(v_2)\ f(v_{k-1}) + \ldots + f(v_k)\ f(v_1)) \bmod 954$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) = зелений, 8892

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-12, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+7, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-12.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 55

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального часу, витраченого на розгляд законів.

Задається інтервал часу. Реалізувати алгоритм впорядкування партій за успішністю за цей проміжок часу – рейтинг успішності зростає для кожного закону за час, коли діють закони, які партія підтримує, і не діють закони, яким вона протистоїть, і навпаки, рейтинг падає, коли діють закони, яким партія протистоїть, і не діють закони, які партія підтримує.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! + n - 1) \bmod 155$
2. Для від’ємних цілих чисел n, $f(n) = n^5 \bmod 255$
3. Для дійсних чисел d, $f(d) = [1/\sin(77d)] \bmod 355$
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)*(f(a)+1) \bmod 555$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)+ f(v_2)+\ldots+ f(v_k)) \bmod 855$

7. Для всіх інших значень f(x) = 8453

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N+13, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+9, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-13.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 56

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, а також мають однакові типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної вартості, витраченої на виправлення дефектів.

Задається тип компонентів. Реалізувати алгоритм, який для кожного компонента розраховує кількість підкомпонентів заданого типу (підкомпоненти довільного рівня вкладеності)

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+156)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^5 - 256|$
3. Для дійсних чисел d, $f(d) = d/(d-356)$
4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(556f(a))}$

6. Для списку v=[v_1,…,v_k], f(v) = max sin (v_i -656)
7. Для всіх інших значень f(x) = 17.56

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+14,деструктор Alpha S:=S-N+13, деструктор Beta S:=S+N, деструктор Gamma S:=S+2N-14.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 57

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов'язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості оцінювань.

Реалізувати алгоритм пошуку найкоротшого шляху через друзів між двома заданими користувачами.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n! \bmod 157$
2. Для від'ємних цілих чисел n, $f(n) = n^3 \bmod 257$
3. Для дійсних чисел d, $f(d) = [\exp(1/\sin(d))] \bmod 357$
4. Для текстових рядків s, f(s) = кількість голосних літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{f(a)} \bmod 557$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_2) + f(v_2)\, f(v_3) + \ldots + f(v_{k-1})\, f(v_k)) \bmod 757$
7. Для всіх інших значень f(x) = 7125

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=S/2-3N+15, деструктор Alpha S:=S+N, деструктор Beta S:=S-2N, деструктор Gamma S:=S+N-13.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 58

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості балів (набраних всіма студентами за усі виконані завдання).

Два студенти називаються «колегами», якщо в якийсь день вони одночасно працювали над певним завданням. Реалізувати алгоритм пошуку всіх студентів, які мають задану кількість колег відносно заданого завдання.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+58)$
2. Для від'ємних цілих чисел n, $f(n) = \log_2 |n^3 + 258|$
3. Для дійсних чисел d, $f(d) = d/(d+358)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(516f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \cos (v_i - 758)$
7. Для всіх інших значень f(x) = 83.58

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого

створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-16, деструктор Alpha S:=S-N, деструктор Beta S:=S+4N+16, деструктор Red S:=S+N/4, деструктор Green S:=S-N/2-8.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 59

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної ваги перевезених вантажів.

Для заданого моменту часу, заданого вантажу, який знаходиться на заданому складі, реалізувати алгоритм створення списку транспортних засобів, впорядкованих відповідно до часу, потрібного їм для того, щоб забрати вантаж зі складу. Кожен транспортний засіб має завершити ту дію, яку він зараз виконує (пересування до наступного складу, завантаження чи розвантаження). Потім, якщо транспортний засіб не зможе взяти вантаж – він має продовжувати рух за своїм розкладом, при цьому не завантажуючи нові вантажі, поки не звільниться достатньо місця (за вагою та об'ємом). Нарешті, транспортний засіб має дістатись заданого складу. Якщо транспортний засіб взагалі не зможе перевозити вантаж (не вистачить ваги чи об'єму навіть без інших вантажів) – такі транспортні засоби додаються наприкінці списку.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) = n^2 mod 159 англійською мовою
2. Для від'ємних цілих чисел n, f(n) = (n^5+n^3) mod 259 англійською мовою
3. Для дійсних чисел d, f(d) = sin(d+359) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)

6. Для списку v=[v_1,…,v_k], f(v) = унікальні слова з усіх f(v_i), у алфавітному порядку
7. Для всіх інших значень f(x) = “sorry try again please”

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може повторюватись між різними базовими класами, починається з 1 для першого створеного екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N-17, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+17, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-7.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 60

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники можуть переходити на інші проекти посередині роботи над проектом (якщо проект, з якого вони переходять, може виконуватись і без них), але не можуть приєднуватись до проекту, якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості “людино-днів”, витрачених на проекти.

Розробник A може тренувати розробника B (який в цьому випадку називається учнем A), якщо вони знають хоча б одну спільну технологію, і при цьому ефективність A більша за B. Реалізувати алгоритм, який починаючи із заданого розробника, виводить всіх учнів цього розробника, потім всіх учнів цих учнів і т.д.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 160
2. Для від’ємних цілих чисел n, $f(n) = (n^4 + n^3) \bmod 260$ – цифри у представленні з основою 4
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 360$, k = 1,2,…, 39

4. Для текстових рядків s, $f(s) = k^k \bmod 560$, де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)
5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(b), повторений кількість разів відповідного елементу зі списку f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = об'єднання обернених списків $f(v_i)$ ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [4,0,7, 0, 8]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+18,деструктор Alpha S:=S-N+18, деструктор Beta S:=S+N, деструктор Gamma S:=S+3N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об'єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об'єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об'єктів.
3. Для заданої кількості об'єктів M (що включає як незалежні, так і вкладені об'єкти) перебрати всі можливі комбінації об'єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 61

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального розміру повністю отриманих повідомлень.

Для заданого проміжку часу та списку типів повідомлень, реалізувати алгоритм впорядкування серверів за кількістю повідомлень одного з цих типів, які сервер отримав чи надіслав за цей проміжок часу.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(2^n + n^2) \bmod 161$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 261$
3. Для дійсних чисел d, f(d) = синій, $[\ 1/\sin(\log_2 d)] \bmod 361$
4. Для текстових рядків s, f(s) = зелений, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 4 до 7 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 661$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\ f(v_k) + f(v_2)\ f(v_{k-1}) + \ldots + f(v_k)\ f(v_1)) \bmod 761$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) =червоний, 8359

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S+N-19, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+5, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-4.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 62

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального часу, витраченого на розгляд законів.

Партії можуть об’єднуватись в блоки, якщо вони мають схожі позиції щодо законів. Задається певний рівень толерантності (ціле число), обчислюється кількість законів, які дві партії одночасно підтримують чи протистоять, від нього віднімається кількість законів, які одна з партій підтримує, а інша протистоїть, якщо сума перевищує рівень толерантності – партії можуть об’єднатись. Для заданого рівня толерантності і двох заданих партій, реалізувати алгоритм пошуку мінімальної кількості проміжних партій, так що задані партії можуть об’єднатись через задані проміжні партії (алгоритм має повідомити, якщо для заданого рівня толерантності об’єднання взагалі неможливе)

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! + n - 1) \bmod 162$
2. Для від’ємних цілих чисел n, $f(n) = n^5 \bmod 262$

3. Для дійсних чисел d, f(d) = [1/sin(87d)] mod 362
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s
5. Для пари p=(a,b), f(p) = f(b)*(f(a)+1) mod 562
6. Для списку v=[v_1,…,v_k], f(v) = (f(v_1)+ f(v_2)+…+ f(v_k)) mod 862
7. Для всіх інших значень f(x) = 8002

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=3S+N+20, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+5, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-20.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 63

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, але можуть мати різні типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної вартості, витраченої на виправлення дефектів.

Реалізувати алгоритм, який для заданого типу дефекту розраховує збитки у винагороді залежно від того, в якому компоненті станеться цей дефект (розглядаються лише компоненти правильного типу, збитки за один період часу без перевірки/виправлення/заміни). Алгоритм має повертати список компонентів, впорядкований за збитками (від більших до менших).

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+163)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^7 + 263|$
3. Для дійсних чисел d, $f(d) = d/(d+363)$

4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(563f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = max sin (v_i -763)
7. Для всіх інших значень f(x) = 12.63

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+21,деструктор Alpha S:=S-N+5, деструктор Beta S:=S+N, деструктор Gamma S:=S+2N-21.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 64

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов'язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості оцінювань.

Для заданої теми, реалізувати алгоритм впорядкування користувачів відповідно до активності в цій темі. Користувачі отримують фіксовану кількість балів за різні види активності, пов'язаної з даною темою – створення нових повідомлень, відповідь на повідомлення, пересилання повідомлення, зміна рейтингу повідомлень. Список будується за загальною кількістю балів, від більшої до меншої.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n! \bmod 164$
2. Для від'ємних цілих чисел n, $f(n) = n^3 \bmod 264$
3. Для дійсних чисел d, $f(d) = [\exp(1/\sin(d))] \bmod 364$
4. Для текстових рядків s, f(s) = кількість голосних літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{f(a)} \bmod 564$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_2) + f(v_2)\, f(v_3) + \ldots + f(v_{k-1})\, f(v_k)) \bmod 764$
7. Для всіх інших значень f(x) = 1642

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори

класів змінюють глобальну змінну S: деструктор Base S:=S/2-2N+22, деструктор Alpha S:=S+3N, деструктор Beta S:=S-2N, деструктор Gamma S:=S+N-22.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 65

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості балів (набраних всіма студентами за усі виконані завдання).

Задається період часу та список знань. Реалізувати алгоритм створення списку студентів, які за цей період підвищили рівень цих знань, впорядкований за сумарним зростанням рівня знань (від більшого до меншого).

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \sin(5n+165)$
2. Для від'ємних цілих чисел n, $f(n) = \log_2 |n^5 + 265)|$
3. Для дійсних чисел d, $f(d) = d/(d+365)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(565f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \cos (v_i - 865)$
7. Для всіх інших значень f(x) = 35.65

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого

створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-23, деструктор Alpha S:=S-N, деструктор Beta S:=S+3N+23, деструктор Red S:=S+N/2, деструктор Green S:=S-N/2-11.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 66

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної ваги перевезених вантажів.

Наземним транспортним засобам забороняється відвідувати склади, з яких за заданий період часу відправляли вантажі заданого типу (не можна навіть проїздити повз такі склади, не зупиняючись на них). Транспортні засоби, які на закінчення цього періоду часу знаходились на одному з цих складів, або їхали з них чи до них, або везли вантаж цього типу, вилучаються з системи. Реалізувати алгоритм знаходження груп складів, між якими тепер неможливо проїхати наземним транспортом, а також кількість транспортних засобів (окремо наземних та повітряних), які залишились в кожній групі складів.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n^2 \bmod 166$ англійською мовою
2. Для від'ємних цілих чисел n, $f(n) = (n^7+n^3) \bmod 266$ англійською мовою
3. Для дійсних чисел d, f(d) = sin(d+366) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = унікальні слова з усіх $f(v_i)$, у алфавітному порядку
7. Для всіх інших значень f(x) = "no luck this time"

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може повторюватись між різними базовими класами, починається з 1 для першого створеного екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N-24, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+24, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-9.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 67

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники не можуть переходити на інші проекти посередині роботи над проектом, і не можуть приєднуватись до проекту, якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості “людино-днів”, витрачених на проекти.

Реалізувати алгоритм побудови списку технологій, впорядкованих за наступними критеріями: 1) кількість пов’язаних проектів; 2) кількість розробників, які знають цю технологію; 3) загальний час виконання проектів, пов’язаних з технологією. Будується спільне значення, що об’єднує ці критерії з певними ваговими коефіцієнтами, і технології впорядковуються за цим значенням.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 167
2. Для від’ємних цілих чисел n, $f(n) = (n^4 + n^3) \bmod 267$ – цифри у представленні з основою 7
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 367$, k = 1,2,…, 42

4. Для текстових рядків s, $f(s) = k^k \bmod 467$, де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)
5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(b), повторений кількість разів відповідного елементу зі списку f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = об'єднання обернених списків $f(v_i)$ ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [4,0,25]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+25,деструктор Alpha S:=S-N, деструктор Beta S:=S+N+25, деструктор Gamma S:=S+3N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об'єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об'єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об'єктів.
3. Для заданої кількості об'єктів M (що включає як незалежні, так і вкладені об'єкти) перебрати всі можливі комбінації об'єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 68

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального розміру повністю отриманих повідомлень.

Адміністратори мережі мають можливість прокласти більш сучасні надшвидкі зв'язки між окремими серверами (лише там, де вже існували зв'язки раніше). Адміністратори хочуть обрати такі зв'язки, щоб 1) загальна кількість нових зв'язків була мінімальною, але при цьому між будь-якими серверами можна було доставити повідомлення лише користуючись новими зв'язками; 2) сумарна кількість повідомлень, яка проходила раніше по цим зв'язкам, була максимальною. Реалізувати алгоритм вибору зв'язків для покращення відповідно до цих умов.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(5^n + n^5) \bmod 168$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 268$
3. Для дійсних чисел d, f(d) = синій, $[1/\sin(\log_2 d)] \bmod 368$
4. Для текстових рядків s, f(s) = зелений, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 2 до 6 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 568$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох

6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_k) + f(v_2)\, f(v_{k-1}) + \ldots + f(v_k)\, f(v_1)) \bmod 768$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) =червоний, 9686

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S+N-26, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+26, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-14.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати $M<7$.

Варіант 69

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального часу, витраченого на розгляд законів.

Реалізувати алгоритм побудови списку законів, впорядкованих за наступними критеріями (у всіх критеріях спершу більше): 1) кількість виборців, які їх підтримують; 2) кількість партій, які їх підтримують; 3) кількість виборців, які їм протистоять; 4) кількість партій, які їм протистоять; 5) кількість розглядів закону в парламенті. Будується спільне значення, що об’єднує ці критерії з певними ваговими коефіцієнтами, і закони впорядковуються за цим значенням.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) = (n! + n -1) mod 169
2. Для від’ємних цілих чисел n, f(n) = n^5 mod 269
3. Для дійсних чисел d, f(d) = [1/sin(77d)] mod 369
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s

5. Для пари p=(a,b), f(p) = f(b)*(f(a)+3) mod 569
6. Для списку v=[v_1,…,v_k], f(v) = (f(v_1)+ f(v_2)+…+ f(v_k)) mod 869
7. Для всіх інших значень f(x) = 7487

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=3S+N+27, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+5, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-27.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 70

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, а також мають однакові типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної вартості, витраченої на виправлення дефектів.

Реалізувати алгоритм, який заданої функції повертає список всіх компонентів, які її виконують. Спершу в списку компоненти найвищого рівня, потім їх підкомпоненти, потім підкомпоненти цих підкомпонентів і т.д.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(7n+170)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^7 + 270|$
3. Для дійсних чисел d, $f(d) = d/(d-570)$
4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(570f(a))}$

6. Для списку v=[v_1,…,v_k], f(v) = max sin (v_i -670)
7. Для всіх інших значень f(x) = 12.707

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+28,деструктор Alpha S:=S-N+13, деструктор Beta S:=S+N-28, деструктор Gamma S:=S+2N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 71

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов'язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості оцінювань.

Для заданого користувача задана тема стає «забороненою» – цей користувач перестає спілкуватись з тими, хто пов'язаний з цією темою (тобто має її серед улюблених, або створював повідомлення на цю тему). Реалізувати алгоритм, який будує список користувачів, з якими даний користувач ще може користуватись – спершу друзі, потім друзі друзів і т.д., наприкінці списку всі користувачі, хто не пов'язані з даним користувачем через друзів (але все одно можуть спілкуватись, тобто не пов'язані із забороненою темою).

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n! \bmod 171$
2. Для від'ємних цілих чисел n, $f(n) = n^3 \bmod 271$
3. Для дійсних чисел d, $f(d) = [\exp(1/\sin(d))] \bmod 471$
4. Для текстових рядків s, f(s) = кількість голосних літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{f(a)} \bmod 571$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_2) + f(v_2)\, f(v_3) + \ldots + f(v_{k-1})\, f(v_k)) \bmod 771$
7. Для всіх інших значень f(x) = 1799

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для

кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=S/2-3N+29, деструктор Alpha S:=S+N, деструктор Beta S:=S-2N-17, деструктор Gamma S:=S+N-29.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 72

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості балів (набраних всіма студентами за усі виконані завдання).

Для двох заданих знань, їх близькість – це кількість завдань, які одночасно пов'язані з обома знаннями. Реалізувати алгоритм побудови дерева знань, так щоб сумарна близькість між сусідніми знаннями дерева була максимальною.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \sin(5n+172)$
2. Для від'ємних цілих чисел n, $f(n) = \log_2 |n^7 - 272|$
3. Для дійсних чисел d, $f(d) = d/(d+472)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(572f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \min \cos (v_i - 772)$
7. Для всіх інших значень f(x) = 35.72

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого

створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-30, деструктор Alpha S:=S-N, деструктор Beta S:=S+3N+30, деструктор Red S:=S+N/2, деструктор Green S:=S-N/2+14.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 73

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної ваги перевезених вантажів.

Реалізувати алгоритм побудови списку транспортних засобів (всіх типів), які доставили найбільшу кількість вантажів заданого типу за заданий проміжок часу. Порядок за кількістю вантажів (від більшої до меншої); для однакової кількості за загальною вагою вантажів цього типу; для однакової кількості та ваги за загальним об'ємом.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n^7 \bmod 173$ англійською мовою
2. Для від'ємних цілих чисел n, $f(n) = (n^5+n^3) \bmod 273$ англійською мовою
3. Для дійсних чисел d, f(d) = sin(d+373) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = унікальні слова з усіх $f(v_i)$, у алфавітному порядку
7. Для всіх інших значень f(x) = "sorry maybe try again"

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може повторюватись між різними базовими класами, починається з 1 для першого створеного

екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N-31, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+31, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-11.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 74

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники не можуть переходити на інші проекти посередині роботи над проектом, але можуть приєднуватись до проекту, навіть якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості “людино-днів”, витрачених на проекти.

Реалізувати алгоритм побудови списку технологій, впорядкованих за наступними критеріями: 1) кількість пов’язаних проектів; 2) кількість розробників, які знають цю технологію; 3) загальний час виконання проектів, пов’язаних з технологією. Для кожної технології визначається позиція в кожному з цих списків, і позиція у загальному списку визначається сумою позицій в кожному списку.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 174
2. Для від’ємних цілих чисел n, $f(n) = (n^4 + n^3) \bmod 274$ – цифри у представленні з основою 7
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 374$, k = 1,2,…, 42

4. Для текстових рядків s, $f(s) = k^k \bmod 574$, де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)
5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(a), повторений кількість разів відповідного елементу зі списку f(b)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = об'єднання обернених списків $f(v_i)$ ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [3,0,47]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+32,деструктор Alpha S:=S-N+4, деструктор Beta S:=S+N, деструктор Gamma S:=S+3N-32.

1. Реалізувати відповідні класи, створити і видалити певну кількість об'єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об'єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об'єктів.
3. Для заданої кількості об'єктів M (що включає як незалежні, так і вкладені об'єкти) перебрати всі можливі комбінації об'єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 75

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального розміру повністю отриманих повідомлень.

Для заданої програми та заданого проміжку часу, реалізувати алгоритм побудови списку серверів, через які найчастіше проходили повідомлення від програми та до неї. Список впорядкований за кількістю повідомлень та загальним розміром повідомлень.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(2^n - n^2) \bmod 175$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 275$
3. Для дійсних чисел d, f(d) = синій, $[1/\sin(\log_2 d)] \bmod 375$
4. Для текстових рядків s, f(s) = зелений, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 3 до 5 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 575$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_k) + f(v_2)\, f(v_{k-1}) + \ldots + f(v_k)\, f(v_1)) \bmod 975$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) =червоний, 9583

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S+N-33, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+33, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-24.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 76

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального часу, витраченого на розгляд законів.

Реалізувати алгоритм побудови списку законів, впорядкованих за наступними критеріями (у всіх критеріях спершу більше): 1) кількість виборців, які їх підтримують; 2) кількість партій, які їх підтримують; 3) кількість виборців, які їм протистоять; 4) кількість партій, які їм протистоять; 5) кількість розглядів закону в парламенті. Для кожного закону визначається позиція в кожному з цих списків, і позиція у загальному списку визначається сумою позицій в кожному списку.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) = (n! + n -1) mod 176
2. Для від’ємних цілих чисел n, f(n) = n^5 mod 276
3. Для дійсних чисел d, f(d) = [1/sin(77d)] mod 376
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s

5. Для пари p=(a,b), f(p) = f(b)*(f(a)+1) mod 476
6. Для списку v=[v_1,…,v_k], f(v) = (f(v_1)+ f(v_2)+…+ f(v_k)) mod 776
7. Для всіх інших значень f(x) = 8974

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=3S+N+34, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+34, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-13.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 77

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, а також мають однакові типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної вартості, витраченої на виправлення дефектів.

Реалізувати алгоритм, який будує список дефектів, впорядкованих за часом існування (від виникнення до усунення шляхом виправлення чи заміни компоненту). Порядок від більшого до меншого часу. Дефекти, які на момент створення списку виявлені, але не виправлені, знаходяться на початку списку. Дефекти, які на момент створення списку виникли, але не були виявлені, до списку не потрапляють.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+177)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^5 + 277|$
3. Для дійсних чисел d, $f(d) = d/(2*d+377)$

4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(577f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = max sin (v_i -777)
7. Для всіх інших значень f(x) = 12.77

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+35,деструктор Alpha S:=S-N+35, деструктор Beta S:=S+N, деструктор Gamma S:=S+2N-17.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Варіант 78

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати соціальну мережу. Соціальна мережа складається з *користувачів*, які створюють *повідомлення* (записи, пости) на відомі *теми*. Кожен користувач має список друзів (інших користувачів) і список улюблених тем. Користувач може створити повідомлення, яке пов'язане з однією чи більше темами та може посилатись на інших користувачів. Користувач може оцінити повідомлення інших користувачів, вибравши варіант «подобається» чи «не подобається» – в результаті змінюється загальний рейтинг повідомлення. Користувач може відповісти на повідомлення – при цьому створюється нове повідомлення, яке містить посилання на перше повідомлення. Користувач може переслати (репостити) повідомлення – при цьому створюється копія повідомлення, до якої користувач може додати посилання на додаткових користувачів. Користувачі мають різні стратегії поведінки – наприклад, лише оцінюють повідомлення інших, оцінюють і пересилають, відповідають, створюють нові повідомлення, …

Створити модель мережі з певною кількістю користувачів, тем та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості оцінювань.

Для заданого повідомлення, реалізувати алгоритм пошуку користувачів, які ще не взаємодіяли з ним (тобто не автор, немає посилання на них, не відповідали, не пересилали повідомлення і не оцінювали). Починаючи з користувачів, які взаємодіяли з повідомленням, будуються списки їх друзів, потім друзів друзів і т.д. Порядок користувачів залежить в першу чергу від ступені участі тих, хто взаємодіяли, і вже потім від рівнів дружби – тобто друзі друзів автора (і їх друзі, і т.д.) у списку раніше, ніж друзі тих, хто лише оцінювали повідомлення.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n! \bmod 178$
2. Для від'ємних цілих чисел n, $f(n) = n^3 \bmod 278$
3. Для дійсних чисел d, $f(d) = [\exp(1/\sin(d))] \bmod 378$
4. Для текстових рядків s, f(s) = кількість голосних літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{f(a)} \bmod 578$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_2) + f(v_2)\, f(v_3) + \ldots + f(v_{k-1})\, f(v_k)) \bmod 878$
7. Для всіх інших значень f(x) = 1786

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний

поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=S/2-3N+36, деструктор Alpha S:=S+N, деструктор Beta S:=S-2N, деструктор Gamma S:=S+N-36.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 79

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати процес навчання в університеті. *Студенти* виконують *завдання* з різних *предметів*, покращуючи свої *знання*. Кожен студент має список знань і поточний рівень знань для кожного з них. Кожне завдання пов'язане з одним чи більше знаннями, а також має рівень складності та максимальну кількість балів за його виконання. Під час виконання завдання визначається отримана кількість балів за завдання (від 0 до максимальної), а також час виконання (в днях): ці параметри залежать від рівня складності завдання, рівня знань студента, що виконує завдання, а також певного випадкового фактору. Виконання завдання підвищує рівень відповідних знань студента. Студенти мають обмежену кількість днів на виконання всіх завдань; за один день можна виконувати не більше одного завдання. Студенти можуть мати різні стратегії виконання завдань (наприклад, починати з найпростіших, чи найскладніших, чи тих, що мають найбільшу кількість балів, чи пов'язані з найкращими знаннями студента). Завдання також можуть бути різних типів (результат більше залежить від рівня складності, чи від мінімального рівня знань, чи від середнього рівня знань, чи від випадкового фактору). Студентам не відомий тип завдань – лише складність, кількість балів та знання.

Створити модель системи з певною кількістю студентів, знань та завдань, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості балів (набраних всіма студентами за усі виконані завдання).

Реалізувати алгоритм побудови списку найскладніших завдань. Задаються кілька критеріїв: 1) сума часу, яке витратили всі студенти на виконання цього завдання, 2) загальне зростання знань всіх студентів в результаті виконання цього завдання, 3) рівень складності завдання. Складаються впорядковані списки по кожному з критеріїв. Загальний список будується наступним чином: беруть перший елемент першого списку, потім перший елемент другого, потім перший елемент третього, потім другий елемент першого списку і т.д. (уникаючи повторів).

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \sin(5n+179)$
2. Для від'ємних цілих чисел n, $f(n) = \log_2 (n^5 + 279)$
3. Для дійсних чисел d, $f(d) = d/(d+379)$
4. Для текстових рядків s, f(s) = квадрат частки символів-цифр в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(579f(a))}$
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = \max \cos (v_i - 879)$

7. Для всіх інших значень f(x) = 35.97

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний поміж всіх підкласів, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також вкладений екземпляр класу Base (який може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S+N-37, деструктор Alpha S:=S-N, деструктор Beta S:=S+3N+37, деструктор Red S:=S+N/2, деструктор Green S:=S-N/2-17.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 80

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати систему доставки вантажів. Система складається з *транспортних засобів*, які доставляють *вантажі* між *складами*. Транспортні засоби бувають наземні чи повітряні. Для кожного складу задаються координати (на площині), а також сусідні склади і відстань до них (по вулицях). Наземний транспорт їздить по вулицях між сусідніми складами; повітряний транспорт може літати напряму в будь-який склад. Для кожного вантажу задається його вага та об'єм. Для кожного транспортного засобу задається максимальна вага та об'єм вантажів, а також швидкість руху і час завантаження/розвантаження. Транспортний засіб може перевозити довільну кількість вантажів (якщо вони вміщаються) між довільними складами, на кожному складі розвантажуватись чи завантажуватись повністю чи частково. Склади бувають різних видів: деякі лише отримують вантажі (певних типів), тоді як інші отримують одні вантажі і відправляють інші вантажі з певною періодичністю.

Створити модель системи з певною кількістю транспортних засобів, складів та вантажів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної ваги перевезених вантажів.

Для заданого транспортного засобу і заданого періоду часу, реалізувати алгоритм побудови списків типів вантажу, який даний засіб доставляв найбільше (впорядкований за кількістю та вагою вантажів), та складів, який він відвідував найчастіше (впорядкований за кількістю відвідувань та сумарною вагою вантажів, доставлених до цього складу та з цього складу).

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає текстове значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = n^2 \bmod 180$ англійською мовою
2. Для від'ємних цілих чисел n, $f(n) = (n^5+n^3) \bmod 280$ англійською мовою
3. Для дійсних чисел d, f(d) = sin(d+380) перші дві цифри після коми англійською мовою
4. Для текстових рядків s, f(s) = слова з s у зворотньому порядку, наприклад, f("ab c d")="d c ab"
5. Для пари p=(a,b), f(p) = по черзі по одному слову з f(b) та f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = унікальні слова з усіх $f(v_i)$, у алфавітному порядку
7. Для всіх інших значень f(x) = "sorry please try again"

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний поміж всіх підкласів відповідного базового класу, може

повторюватись між різними базовими класами, починається з 1 для першого створеного екземпляру кожного базового класу, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класів Base1 та Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=2S+N-38, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-N+38, деструктор Beta S:=S+N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-21.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 81

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати IT компанію. Компанія складається з *розробників*, які працюють над *проектами* з використанням *технологій*. Кожен розробник знає певний список технологій, а також має загальний рівень ефективності роботи. Кожен проект вимагає використання певного списку технологій і має загальний рівень складності. Кожен проект також має список (можливо, пустий) залежностей – інших проектів, робота над якими має завершитись до початку роботи над даним проектом. Час виконання проекту залежить від його складності, кількості розробників та їх ефективності, а також випадкових факторів. Лише розробники, що знають хоча б одну з потрібних технологій, можуть працювати над проектом. Для кожної технології, потрібної для проекту, має бути хоча б один розробник, який її знає та працює над проектом. Кожен розробник може працювати лише над одним проектом одночасно, але компанія в цілому може працювати над довільною кількістю проектів одночасно (якщо вимоги всіх проектів виконані). Розробники можуть переходити на інші проекти посередині роботи над проектом (якщо проект, з якого вони переходять, може виконуватись і без них), але не можуть приєднуватись до проекту, якщо вони не працювали над ним з початку. Проекти бувають різних видів – з різною залежністю часу виконання від всіх факторів. Технології також бувають різних видів – деякі мають шанс збільшити чи навпаки скоротити час виконання відповідних проектів, деякі збільшують чи зменшують час виконання, якщо в проекті багато чи мало розробників знають цю технологію, і т.д.

Створити модель компанії з певною кількістю розробників, технологій та проектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної кількості “людино-днів”, витрачених на проекти.

Для двох заданих технологій, їх близькість – це кількість проектів, які одночасно їх використовують плюс кількість розробників, які одночасно ними володіють. Реалізувати алгоритм побудови дерева технологій, так щоб сумарна близькість між сусідніми технологіями дерева була максимальною.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає список цілих чисел за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = (n! - n^2) \bmod k$, k=2,3,…, 181
2. Для від’ємних цілих чисел n, $f(n) = (n^4 + n^3) \bmod 281$ – цифри у представленні з основою 4
3. Для дійсних чисел d, $f(d) = [\exp(1/\cos(k*d))] \bmod 481$, k = 1,2,…, 42

4. Для текстових рядків s, $f(s) = k^k \bmod 881$, де k – послідовні двозначні числа з рядку s (розділені довільними нецифровими символами)
5. Для пари p=(a,b), f(p) = кожен елемент зі списку f(b), повторений кількість разів відповідного елементу зі списку f(a)
6. Для списку $v=[v_1,\ldots,v_k]$, f(v) = об'єднання обернених списків $f(v_i)$ ([1,2],[3,4,5]=>[2,1,5,4,3])
7. Для всіх інших значень f(x) = [4,0,4,1]

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=2S-N+39,деструктор Alpha S:=S-N+39, деструктор Beta S:=S+N, деструктор Gamma S:=S+3N.

1. Реалізувати відповідні класи, створити і видалити певну кількість об'єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об'єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об'єктів.
3. Для заданої кількості об'єктів M (що включає як незалежні, так і вкладені об'єкти) перебрати всі можливі комбінації об'єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 82

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати обчислювальну мережу. Мережа складається з *серверів*, на яких запущені *програми*, що обмінюються *повідомленнями*. Кожне повідомлення містить інформацію про відправника та отримувача (в обох випадках сервер та програма), а також тип та розмір (повідомлення більшого розміру передаються довше при однаковій швидкості зв'язку). Кожен сервер містить список сусідніх серверів; повідомлення можуть пересилатись лише на сусідні сервери, тому для доставки отримувачу може виникнути необхідність пересилати повідомлення через кілька серверів. Для кожної пари сусідніх серверів задається швидкість передачі повідомлень між ними. Програми бувають різних типів: деякі лише надсилають повідомлення, деякі лише отримують, інші і надсилають і отримують; деякі програми надсилають повідомлення з певною періодичністю, інші випадково, інші надсилають повідомлення через деякий час після отримання інших повідомлень; деякі програми чекають на повідомлення конкретних типів, щоб продовжити роботу.

Створити модель мережі з певною кількістю серверів, програм та повідомлень, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального розміру повністю отриманих повідомлень.

Для заданого списку типів повідомлень, реалізувати алгоритм побудови 1) списку серверів, через які найчастіше пересилались повідомлення цих типів, та 2) списку програм, які найчастіше відправляли чи отримували повідомлення цих типів. Списки впорядковані за кількістю повідомлень та сумарним розміром повідомлень.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає пару (колір, ціле значення) за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) =червоний, $(2^n + n^2) \bmod 182$
2. Для від'ємних цілих чисел n, f(n) = зелений, $(n^5 + n - 1) \bmod 282$
3. Для дійсних чисел d, f(d) = синій, $[\ 1/\sin(\log_2 d)] \bmod 382$
4. Для текстових рядків s, f(s) = червоний, кількість слів (довільних послідовностей літер англійської мови, розділених пробілами) довжини від 2 до 4 в рядку s
5. Для пари p=(a,b), $f(p) = f(a)^{f(b)} \bmod 582$, якщо кольори однакові – той самий колір, якщо різні – колір, якого нема серед цих двох
6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)\, f(v_k) + f(v_2)\, f(v_{k-1}) + \ldots + f(v_k)\, f(v_1)) \bmod 982$, колір, що зустрічається найчастіше, якщо таких кілька – той з них, що зустрічається останнім
7. Для всіх інших значень f(x) = синій, 3890

Завдання 3 (20% балів+бонус). Задано абстрактний базовий клас Base, два його підкласи Alpha, Beta, і два підкласи Alpha – Red, Green. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також два вкладені екземпляри класу Base (кожен з яких може бути відсутнім). Деструктори класів змінюють глобальну змінну S: деструктор Base S:=4S+N-40, деструктор Alpha S:=S/3-N, деструктор Beta S:=S+2N+23, деструктор Red S:=S-N/2, деструктор Green S:=S-N/2-40.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<7.

Варіант 83

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати демократичну систему правління. Система складається з *виборців*, які голосують за *партії* , що підтримують *закони*. Кожен виборець підтримує певний список законів і протистоїть іншому списку законів. Кожна партія також підтримує певний список законів і протистоїть іншому списку законів; прийшовши до влади, партія намагається прийняти закони, які вона підтримує, і відхилити ті, яким вона протистоїть. Партії, які прийшли до влади, утворюють парламент, де розглядаються закони. В кожен момент часу, парламент може розглядати лише один закон. Кожен закон має складність, яка впливає на час розгляду. Деякі з законів описують власне процес виборів (як часто вони проводяться, як проходить голосування, які партії проходять до парламенту, скільки представників в парламенті від кожної партії). Деякі закони описують процес роботи парламенту (як довго розглядають закони, скільки потрібно голосів для прийняття чи відхилення). Існують також інші закони, які в даній моделі не впливають на її функціонування, але є предметом підтримки виборців та партій. Виборці мають різні стратегії голосування: хтось голосує за критеріями підтримки чи не підтримки своїх законів, хтось голосує за партії, які зараз при владі чи не при владі, хтось голосує випадковим чином. Партії мають різні стратегії розгляду законів: починають приймати підтримувані закони чи відхиляти непідтримувані, починають з простіших чи складніших законів, вибирають закони випадковим чином.

Створити модель системи з певною кількістю виборців, партій та законів, промоделювати її роботу протягом певного часу, а також до досягнення заданого загального часу, витраченого на розгляд законів.

Задається інтервал часу. Реалізувати алгоритм впорядкування виборців за рівнем задоволення діями партій за цей проміжок часу. Рівень задоволення зростає для кожного закону за час, коли діють закони, які підтримує цей виборець і партія, за яку він голосував, і не діють закони, яким вони протистоять, і навпаки, рівень задоволення падає, коли діють закони, яким виборець та його партія протистоять, і не діють закони, які вони підтримують.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає ціле значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, f(n) = (n! + n -3) mod 183
2. Для від’ємних цілих чисел n, f(n) = n^3 mod 283
3. Для дійсних чисел d, f(d) = [1/sin(77d)] mod 383
4. Для текстових рядків s, f(s) = кількість великих літер англійської мови в рядку s
5. Для пари p=(a,b), f(p) = f(b)*(f(a)+1) mod 583

6. Для списку $v=[v_1,\ldots,v_k]$, $f(v) = (f(v_1)+ f(v_2)+\ldots+ f(v_k)) \bmod 783$
7. Для всіх інших значень $f(x) = 8983$

Завдання 3 (20% балів+бонус). Задано абстрактні базові клас Base1, Base2, і підкласи Base1: Alpha, Beta та Base2: Gamma, Delta. Кожен екземпляр класів Base1, Base2 містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру, зростає на 1 для кожного наступного екземпляру), а також по одному вкладеному екземпляру класу Base1 та по два екземпляри класу Base2. Деструктори класів змінюють глобальну змінну S: деструктор Base1 S:=3S+N+41, деструктор Base2 S:=S/2-N, деструктор Alpha S:=S-2N+14, деструктор Beta S:=S-N, деструктор Gamma S:=S-N, деструктор Delta S:=S+3N-41.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати $M<5$.

Варіант 84

Завдання 1 (60% балів). Використовуючи засоби ООП, змоделювати складну систему. Система складається з *компонентів*, які виконують *функції* та в яких можуть виникати *дефекти*. Кожен компонент має тип, вартість, список виконуваних функцій, а також список підкомпонентів (можливо, пустий). Компоненти однакового типу виконують однакові функції, а також мають однакові типи підкомпонентів. Дефекти виникають в компонентах певного типу, і перешкоджають виконанню функцій компоненту (всіх або частини). За кожен період часу, коли система виконує кожну функцію, вона отримує винагороду. Для деяких функцій необхідно, щоб всі компоненти з цією функцією працювали без дефектів; для інших досить певної заданої кількості працюючих компонентів. В системі немає можливості одразу помітити, в яких компонентах виникли які дефекти. Дефекти можна помітити лише коли система перестає виконувати певні функції. В цьому випадку є можливість зупинити деякі компоненти для перевірки. Компоненти, які зупинили для перевірки, не виконують жодної зі своїх функцій протягом перевірки, і всі їх підкомпоненти також не працюють. За кожен період часу, коли компонент перевіряється, є шанс знайти дефект (цей шанс визначається типом дефекту). Коли дефект ідентифіковано, можна витратити певну кількість періодів часу на його виправлення. Кількість часу на виправлення та вартість виправлення залежить від типу компоненту та типу дефекту. В будь-який момент часу можна також замінити певний компонент на новий такого ж типу (підкомпоненти при цьому не заміняються, а переходять від старого до нового компоненту) – на це витрачається один період часу і сплачується повна вартість нового компоненту.

Створити модель системи з певною кількістю компонентів, функцій та дефектів, промоделювати її роботу протягом певного часу, а також до досягнення заданої загальної вартості, витраченої на виправлення дефектів.

Реалізувати алгоритм, який для даної функції виводить список компонентів, які її виконують. Список починається з компонентів найвищого рівня, потім їх підкомпоненти, потім підкомпоненти цих підкомпонентів і т.д.

Завдання 2 (20% балів). Реалізувати функцію f(x), що повертає дійсне значення за наступними правилами:

1. Для натуральних (цілих додатніх) чисел n, $f(n) = \cos(5n+184)$
2. Для від’ємних цілих чисел n, $f(n) = \log_{10} |n^5 + 284|$
3. Для дійсних чисел d, $f(d) = d/(d+384)$
4. Для текстових рядків s, f(s) = куб частки символів-літер англійської мови в рядку s
5. Для пари p=(a,b), $f(p) = f(b)^{\sin(584f(a))}$

6. Для списку v=[v_1,…,v_k], f(v) = max sin (v_i -784)
7. Для всіх інших значень f(x) = 84.15i

Завдання 3 (20% балів+бонус). Задано базовий клас Base (не абстрактний), і три його підкласи Alpha, Beta, Gamma. Кожен екземпляр класу Base містить порядковий номер N (унікальний для кожного підкласу, починається з 1 для першого створеного екземпляру даного підкласу, зростає на 1 для кожного наступного екземпляру), а також вкладений список екземплярів класу Base. Деструктори класів змінюють глобальну змінну S: деструктор Base S:=3S-N+42,деструктор Alpha S:=S-N+3, деструктор Beta S:=S+N, деструктор Gamma S:=S+3N-42.

1. Реалізувати відповідні класи, створити і видалити певну кількість об’єктів, вивести остаточне значення глобальної змінної.
2. Реалізувати функцію, яка буде отримувати список об’єктів і передбачати значення глобальної змінної після послідовного видалення всіх цих об’єктів.
3. Для заданої кількості об’єктів M (що включає як незалежні, так і вкладені об’єкти) перебрати всі можливі комбінації об’єктів, і вивести для кожного остаточне значення змінної. Можна вважати M<5.

Лабораторна робота № 8.3. Індивідуальні та командні проекти

Мета роботи

Навчитися створювати цілісні проекти, використовуючи сучасні підходи до програмування. Для командних проектів – навчитися працювати в команді.

Завдання № 1. Моделювання з використанням UML

Загальні рекомендації

Метою виконання Завдання №1 є вивчення засобів проектування та моделювання програмних систем, зокрема з використанням мови UML. Лабораторна складається з основного завдання (описаного в цьому файлі), а також трьох додаткових завдань (додаткові завдання №№ 1, 2). Виконання додаткових завдань не є обов'язковим, проте дозволить отримати ***додаткові бали***.

Для цієї лабораторної треба викладати в репозиторій не лише код, а й діаграми UML. Необхідно викладати як файли проектів (відповідно до інструменту, який використовується для розробки діаграм), так і згенеровані з них зображення.

Design/Implementation Modeling

Використати UML діаграми для опису структури існуючого коду та його рефакторінгу. Виконання цього завдання складається з наступних кроків:

1. Обрати існуючу програму/проект, з якою планується працювати. Це може бути навчальний проект з минулого семестру, чи одна з лабораторних робіт, чи код з інших предметів, чи з якихось онлайн курсів, чи щось подібне. Також це може бути невеличкий open-source проект. Можна взяти кілька невеликих програм з метою їх подальшого об'єднання. Код має бути досить складним – тобто не рівня Hello world чи реалізації одного нескладного алгоритму (хоча це може бути кілька схожих чи якось пов'язаних алгоритмів, і на подальших кроках можна буде створити для них спільний програмний інтерфейс).
2. Реалізувати unit tests, шо описують функціональність обраної програми. (Якщо такі тести вже існують – їх можна доповнити, або залишити як є)
3. Побудувати UML діаграми, що описують обрану програму. Варто описати сценарії використання (Use Case), структуру коду (Class, Component, Object, Composite Structure, Deployment, Package, Profile), логіку та поведінку програми (Sequence,

Communication, Timing, Activity, Interaction Overview, State). Для побудови деяких діаграм можна використати автоматичну генерацію діаграм з коду; але при цьому діаграми мають бути зрозумілими. Наприклад, взяти 100 класів і кинути їх усі на одну діаграму класів – мабуть, не найкращий варіант ☺

4. Запропонувати якісь зміни в структурі/інтерфейсі/реалізації програми. Це може бути кращий object-oriented design, кращий поділ на компоненти чи відокремлення різних аспектів (наприклад, логіки програми від графічного інтерфейсу), використання якихось патернів проектування, можливість вибору різних варіантів реалізації і т.д. Бажано використовувати побудовану модель програми для опису запропонованих змін. Запропоновані зміни треба узгодити з викладачем.
5. Реалізувати запропоновані зміни.
6. Перевірити, що нова версія програми не вносить зміни в логіку/алгоритми (якщо це не було заплановано). Використати для цього реалізовані раніше unit tests і аналогічні тести, які будуть реалізовані для нової версії.
7. Порівняти попередню та оновлену версії програми за часом виконання окремих алгоритмів/функцій, обсягом коду і т.д.

Мета цього завдання– покращити структуру коду, зробити його більш гнучким та розширюваним. Як частину перетворень, можна реалізувати нову функціональність чи виправити недоліки в попередній – але це має бути додатково до покращень object-oriented design, а не замість нього.

У виконаному завдані має бути чітко зрозуміло, яким був попередній код (до змін/рефакторінгу), та які зміни було зроблено. Наприклад, можна викласти існуючий код в репозиторій першим комітом, помітити його тегом, і потім робити зміни.

Моделі можуть складатись з текстових описів та/або UML діаграм. Також має бути зрозуміло, де модель попереднього коду, а де запропоновані зміни, описані на рівні моделі. Не треба будувати модель/діаграми двічі (для попереднього та зміненого коду) – краще описати попередній код, і потім лише ті зміни, які пропонується зробити.

Завдання № 2. Проектування та розробка програм з використанням патернів проектування

Розробити програму для демонстрації роботи певного класу алгоритмів. Програма має реалізовувати графічний інтерфейс користувача (наприклад, з використанням Qt або інших фреймворків), збереження даних (наприклад, в файли чи базу даних).

Цю лабораторну роботу бажано виконувати в командах з 3-4 студентів (команди можуть бути такими ж, як для проектів, або відрізнятись.) Програма має бути спроектована таким чином, щоб був поділ на відносно незалежні компоненти, і можливість різним учасникам команди працювати над різними компонентами незалежно. В репозиторії має бути видно, ким був написаний кожен фрагмент коду (варіант «один з команди комітить за всіх» не приймається).

Можливі ідеї для компонентів:

1. Власне реалізації алгоритмів, структур даних.
2. Механізм виміру продуктивності алгоритмів (час виконання, обсяг пам’яті)
3. Механізм оцінки теоретичної складності алгоритмів.
4. Механізм підбору параметрів алгоритмів з метою підвищення продуктивності.
5. Візуалізація поведінки алгоритмів
6. Візуалізація теоретичної та практичної продуктивності алгоритмів (таблиці, графіки, …)
7. Документація щодо алгоритмів
8. Інші можливі компоненти, які мають сенс для цієї задачі

Можна розбити програму на дві частини:

1) бібліотеку з реалізацією алгоритмів чи структур даних та
2) застосунок з графічним інтерфейсом, який буде використовувати написану бібліотеку.

Необхідно підготувати документацію реалізованої програми. Зокрема, треба описати заплановану архітектуру програми (використовуючи текстові описи та/або UML діаграми). Також має бути опис того, як використовувати можливості програми – як через графічний інтерфейс, так і з коду.

Необхідно реалізувати юніт-тести для перевірки реалізованих алгоритмів. Зокрема, необхідно реалізувати перевірку того, що різні версії алгоритмів (тобто різні версії одного алгоритму) повертають однакові значення, якщо вони так мають працювати. Або перевірити

властивості чи інваріанти структур даних чи алгоритмів (наприклад, що збалансоване дерево має приблизно однакову висоту у всіх гілках).

Юніт-тести мають більш-менш покривати реалізовану функціональність алгоритмів. Робити тести для інтерфейсу користувача, збереження даних в БД і т.д. не обов'язково (хоча можна, для отримання ***додаткових балів***)

Використання патернів проектування

Мета цієї лабораторної роботи – навчитись реалізовувати і використовувати на практиці патерни проектування (design patterns). Тому бажано реалізувати хоча б основні і найбільш популярні патерни.

Ідеї щодо використання патернів:

1. Патерн **Strategy** – варто використати для реалізації алгоритмів, з можливістю заміни реалізації без зміни клієнтського коду
2. Патерн **Template Method** – варто використати для реалізації варіантів алгоритмів, коли загальна структура лишається незмінною, але якісь аспекти реалізуються по-різному в підкласах
3. Патерн **Composite** – варто використати для реалізації складних алгоритмів, які містять під-алгоритми. Можна також використати для реалізації структур даних.
4. Патерн **Decorator** – варто використати для модифікацій алгоритмів, наприклад алгоритми з додатковою перевіркою правильності параметрів. Також варто використати для вимірів часу виконання. Можна використати і для інших модифікацій.
5. Патерн **Iterator**– варто використати для обходу структур даних.
6. Патерн **Adapter**– варто використати з метою використання бібліотечних реалізацій алгоритмів (зі стандартної бібліотеки або сторонніх бібліотек)
7. Патерн **Abstract Factory / Factory Method**– варто використати для побудови стандартних реалізацій певних алгоритмів (залежно від налаштувань)
8. Патерн **Builder**– варто використати для побудови складних алгоритмів, які складаються з багатьох частин
9. Патерн **Singleton**– варто використати для підтримки класів, які мають існувати в одному екземплярі і бути доступними іншим класам (наприклад, класи для побудови алгоритмів)
10. Патерн **Visitor**– варто використати для розрахунку певних властивостей складних алгоритмів/структур даних, наприклад теоретичної складності чи обсягу пам'яті
11. Патерн **Bridge**– варто використати для створення кількох версій інтерфейсу

алгоритму чи структури даних, і незалежного розвитку їх реалізацій

12. Патерн **Command**– варто використати для реалізації інтерактивного режиму роботи з алгоритмами, зокрема можливість виконувати послідовність операцій, а також режим «машини часу» з можливістю повернути назад операції
13. Патерн **Memento**– варто використати для збереження та відновлення стану алгоритмів та структур даних (можливість завершити програму посередині інтерактивного режиму і потім відновитись з тої ж позиції)
14. Патерн **Facade**– варто використати для загального інтерфейсу компонентів та взаємодії компонентів між собою.
15. Інші патерни як з книги GoF, так і з інших джерел – варто використовувати там, де це доцільно.

Додатково до коду необхідно описати використання патернів в документації.

Зазвичай є рекомендація «не зловживати» використанням патернів, тобто не намагатись втиснути в одну програму усі патерни, які відомі розробнику. Для цієї лабораторної з демонстраційною метою можна послабити цю рекомендацію: тобто якщо в деякій ситуації можна реалізувати щось або з патернами, або без патернів – то варто про це написати в документації, і варто реалізувати варіант з патернами. (Можна також реалізувати обидва варіанти і додати механізм вибору між ними; за якісну реалізацію будуть виставлені ***додаткові бали***)

Можливі варіанти

Варіанти цієї лабораторної будуються навколо певного класу близьких алгоритмів. Необхідно реалізувати різні алгоритми з обраного класу, починаючи від простих і до досить складних.

Можливі ідеї варіантів

1. Списки і схожі структури
2. Дерева (в тому числі балансовані дерева пошуку)
3. Індексовані таблиці (реалізації на основі хеш-таблиць та дерев)
4. Сортування
5. Алгоритми на графах
6. Комбінаторні алгоритми (зокрема, з використанням динамічного програмування, backtracking, методу гілок і границь, …)
7. Алгоритми дискретної оптимізації (пошук максимуму/мінімуму)
8. Алгоритми теорії чисел
9. Алгоритми лінійної алгебри (вектори, матриці, лінійні рівняння, …)

10. Алгоритми математичного аналізу (похідні, інтеграли, як символьні, так і чисельні алгоритми)
11. Чисельні методи
12. Алгоритми математичної статистики

В рамках однієї теми можна фокусуватись на різних аспектах алгоритмів. Можна також запропонувати свою тему, але треба погодити її з викладачем.

Додаткове завдання № 1. Моделювання з використанням UML

Загальні рекомендації

Для цього завдання треба викладати в репозиторій не лише код, а й діаграми UML. Необхідно викладати як файли проектів (відповідно до інструменту, який використовується для розробки діаграм), так і згенеровані з них зображення. Обов'язково викладати глосарій (наприклад, в текстовому файлі).

Conceptual/Domain Modeling

Змоделювати з використанням UML роботу користувача з популярними сервісами (конкретний сервіс залежно від варіанту). Створити глосарій та описати предметну область, основні рішення з архітектури та проектування у вигляді UML діаграм. Код писати не потрібно (хоча можна, за бажання).

Бажано описати загалом основну функціональність (діаграми Use Case, Class, Component, Deployment, Package) і ***вибрати кілька (3-5) сценаріїв використання для більш детального опису*** (діаграми Object, Composite Structure, Sequence, Communication, Timing, Activity, Interaction Overview, State, Profile).

Бажано використовувати різноманітні діаграми UML 2.x – щоб зрозуміти, в чому між ними відмінність та де їх використовують. Якщо якийсь з 14 типів діаграм не використовувався – треба пояснити, чому. Для деяких типів діаграм варто реалізувати кілька діаграм одного типу, наприклад для ілюстрації різних сценаріїв використання чи різних компонентів.

Для отримання **додаткових балів** можна подумати, якої функціональності не вистачає в обраному сервісі, чи що можна було б покращити, і спроектувати реалізацію цієї функціональності (з використанням UML діаграм).

Кожен варіант описує конкретний сервіс. Можна обрати аналогічний сервіс замість того, який вказано у списку (наприклад, замість Google Maps можна змоделювати Bing Maps чи OpenStreetMap). Проте не варто моделювати якісь абстрактні сервіси (систему мап взагалі) – краще обрати конкретний сервіс і описувати його.

Варіанти завдань

1. Web search (e.g. Google)
2. Web mail (e.g. GMail)
3. Collaborative document editing (e.g. Google Docs)
4. Cloud file storage and file sharing (e.g. Dropbox)

5. Web maps (e.g. Google Maps)
6. Social network (e.g. Facebook)
7. Messenger (e.g. Telegram, WhatsApp, Viber, Skype, …)
8. Twitter
9. Blog service (e.g. Medium)
10. Code repository (e.g. GitHub, Bitbucket or GitLab)
11. Project management/issue tracking system (e.g. GitHub Issues, Jira, Trac, Bugzilla, …)
12. IDE (e.g. Visual Studio, Eclipse, Qt Creator, …)
13. UML modeling / diagram tool (e.g. StarUML, PlantUML, yEd, …)
14. Instagram
15. Інші аналогічні сервіси, на вибір студентів.

Додаткове завдання № 2. Peer Review (Architecture/Design)

В цьому завданні студенти переглядають моделі (зокрема, UML діаграми), що були реалізовані іншими студентами (зокрема, у завданні №1 та/або додатковому завданні 1), описують свої враження від них і пропонують покращення.

Під час перегляду моделей варто відповісти на такі питання:

1. Наскільки моделі є зрозумілими, наскільки вони описують предметну область, структуру та поведінку відповідної системи?
2. Чи є якісь аспекти, які видаються важливими, але не відображені в моделі (на діаграмах)?
3. Чи є в моделі щось зайве, якісь аспекти описані занадто детально?
4. Наскільки доцільно використані різні типи діаграм?
5. Наскільки коректно використана нотація UML, різні елементи та конектори?
6. Наскільки вдалим є глосарій? Чи всі важливі поняття предметної області описано? Чи немає неоднозначностей?
7. Чи всі важливі сценарії використання описано в моделі? Наскільки зрозумілі різні сценарії, зв'язки між ними?
8. Наскільки доцільним є поділ системи на частини/компоненти/модулі/…?
9. Наскільки доцільними є зв'язки між компонентами/класами/об'єктами? Чи немає занадто тісно зв'язаних компонентів?
10. Наскільки object-oriented design відповідає загальним принципам?

Якщо студенти-reviewers мають якісь позитивні коментарі (тобто щось реалізовано правильно, або використано цікаві механізми UML, або ще щось) – про це варто також писати в review.

Студенти-автори моделей отримають доступ до коментарів студентів, які робили review, і матимуть змогу покращити моделі – чи пояснити, чому вони не погоджуються із знайденими недоліками/запропонованими покращеннями.

Варто зазначити, що студентам-авторам моделей не будуть знижуватись бали за те, що інші студенти знайшли недоліки в їх моделях (за умови, що ці недоліки будуть виправлені). Тому не варто приховувати знайдені проблеми в моделях своїх колег – це не допоможе авторам моделей, а лише знизить бали, отримані за це додаткове завдання студентами-reviewers.

Питання та завдання для контролю знань

S.O.L.I.D.

1. Що таке S.O.L.I.D.?
2. Що таке принцип єдиної відповідальності SRP?
3. Що таке принцип відкритості / закритості OCP?
4. Що таке принцип принцип підстановки Лісков LSP?
5. Що таке принцип принцип відокремлення інтерйесів ISP?
6. Що таке принцип принцип інверсії залежностей DIP?

Предметний покажчик

Література

Основна

1. Павловская Т.А. С/С++. Программирование на языке высокого уровня СПб.: Питер, 2007. – 461 с.
2. Павловская Т.А., Щупак Ю.А. С/С++. Объектно-ориентированное программирование: Практикум СПб.: Питер, 2005. – 265 с.
3. Дейтел Х.М., Дейтел П.Дж. Как программировать на С++ М.: Бином-Пресс, 2005. – 1248 с.
4. Уэллин С. Как не надо программировать на С++ СПб.: Питер, 2004. – 240 с.
5. Хортон А. Visual С++ 2005: Базовый курс М.: Вильямс, 2007. – 1152 с.
6. Солтер Н.А., Клеппер С.Дж. С++ для профессионалов. М.: Вильямс, 2006. – 912 с.
7. Лафоре Р. Объектно-ориентированное программирование в С++ СПб.: Питер, 2006. – 928 с.
8. Лаптев В.В. С++. Объектно-ориентированное программирование СПБ.: Питер, 2008. – 464 с.
9. Лаптев В.В., Морозов А.В., Бокова А.В. С++. Объектно-ориентированное программирование. Задачи и упражнения СПБ.: Питер. – 2007. – 288 с.: ил.
10. https://professorweb.ru/my/it/blog/net/solid.php
11. https://metanit.com/sharp/patterns/5.1.php

Додаткова

12. Прата С. Язык программирования С++. Лекции и упражнения СПб.: ДиаСофт, 2003. – 1104 с.
13. Мейн М., Савитч У. Структуры данных и другие объекты в С++ М.: Вильямс, 2002. – 832 с.
14. Саттер Г. Решение сложных задач на С++ М.: Вильямс, 2003. – 400 с.
15. Чепмен Д. Освой самостоятельно Visual С++ .NET за 21 день М: Вильямс, 2002. – 720 с.
16. Мартынов Н.Н. Программирование для Windows на С/С++ М.: Бином-Пресс, 2004. – 528 с.
17. Паппас К., Мюррей У. Эффективная работа: Visual С++ .NET СПб.: Питер, 2002. – 816 с. М.: Вильямс, 2001. – 832 с.
18. Грэхем И. Объектно-ориентированные методы. Принципы и практика М.: Вильямс, 2004. – 880 с.

19. Элиенс А. Принципы объектно-ориентированной разработки программ М.: Вильямс, 2002. – 496 с.
20. Ларман К. Применение UML и шаблонов проектирования М.: Вильямс, 2002. – 624 с.
21. Шилэт Г. Полный справочник по C. 4-е издание. М.-СПб.-К: Вильямс, 2002.
22. Прата С. Язык программирования C. Лекции и упражнения. М.: ДиаСофтЮП, 2002.
23. Александреску А. Современное проектирование на C++ М.: Вильямс, 2002.
24. Браунси К. Основные концепции структур данных и реализация в C++ М.: Вильямс, 2002.
25. Подбельский В.В. Язык СИ++. Учебное пособие. М.: Финансы и статистика, 2003.
26. Павловская Т.А., Щупак Ю.А. C/C++. Программирование на языке высокого уровня. СПб.: Питер, 2002.
27. Савитч У. Язык C++. Объектно-ориентированного программирования. М.-СПб.-К.: Вильямс, 2001.