МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет
«Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання самостійної роботи
з дисципліни

“Системний аналіз”

для студентів спеціальностей
121 “Інженерія програмного забезпечення” та
122 “Комп’ютерні науки”
(денної форми навчання)

2023

Методичні вказівки до виконання самостійної роботи з дисципліни “Системний аналіз” для студентів спеціальностей 121 “Інженерія програмного забезпечення” та 122 “Комп’ютерні науки” (денної форми навчання) / В.М. Льовкін. – Запоріжжя : НУ «Запорізька політехніка», 2023. – 26 с.

Автор: В.М. Льовкін, канд. техн. наук, доцент

Рецензент: А.О. Олійник, д-р. техн. наук, професор

Відповідальний
за випуск: С.О. Субботін, д-р. техн. наук, професор

Затверджено
на засіданні кафедри
програмних засобів

Протокол № 8
від “30” березня 2023 р.

ЗМІСТ

ВСТУП

Дане видання призначене для вивчення студентами денної форми навчання системного аналізу.

Відповідно до графіка студенти перед виконанням розрахунково-графічного завдання повинні виконати лабораторні роботи та ознайомитися з конспектом лекцій і рекомендованою літературою. Дані методичні вказівки містять тільки основні, базові теоретичні відомості, необхідні для виконання розрахунково-графічного завдання, тому для виконання роботи та при підготовці до її захисту необхідно ознайомитись з конспектом лекцій та опрацювати весь необхідний матеріал, наведений в переліку рекомендованої літературі, використовуючи також статті у інтернет-виданнях і актуальних наукових журналах з комп'ютерних наук.

Для одержання заліку студент повинен у відповідності зі всіма наведеними вимогами розробити програмне забезпечення та оформити звіт, після чого продемонструвати на комп'ютері розроблене програмне забезпечення з виконанням всіх запропонованих викладачем тестів.

Усі завдання повинні виконуватись студентами індивідуально і не містити плагіату як в оформленому звіті так і в розробленому програмному забезпеченні.

Під час співбесіди при захисті роботи студент повинен виявити знання щодо мети роботи, теоретичного матеріалу, методів виконання кожного етапу роботи, змісту основних розділів звіту з демонстрацією результатів на конкретних прикладах, практичних прийомів використання теоретичного матеріалу в розробленому програмному забезпеченні. Студент повинен вміти обґрунтувати всі прийняті ним проєктні рішення і рішення з реалізації програмного забезпечення, аналізу і керування ризиками та правильно аналізувати і використовувати на практиці отримані результати. Для базової самоперевірки при підготовці до виконання і захисту роботи студент повинен відповісти на контрольні запитання, наведені наприкінці опису відповідної роботи.

1 РОЗРАХУНКОВО-ГРАФІЧНЕ ЗАВДАННЯ № 1 ЗАСТОСУВАННЯ СИСТЕМНОГО ПІДХОДУ ДО ПРОГРАМУВАННЯ МОВОЮ PYTHON

1.1 Основні особливості мови програмування Python

Python – популярна високорівнева мова програмування, яка використовується для розроблення програм та створення прикладних сценаріїв у різноманітних галузях.

Python має потужну довідкову систему. Довідкові дані можна отримати як за допомогою документації Python, доступної зокрема з меню IDLE, так і за допомогою відповідних функцій через інтерпретатор.

Функція dir повертає перелік всіх доступних атрибутів заданого об’єкту:

– для того щоб дізнатись про всі інструменти, доступні в деякому бібліотечному модулі, необхідно імпортувати його та передати ім’я даного модуля функції dir;

– для того щоб дізнатись про всі атрибути об’єктів вбудованих типів, необхідно передати функції dir літерал або існуючий об’єкт відповідного типу.

Для того щоб отримати інформацію про призначення того чи іншого методу, необхідно передати його ім’я функції help.

У мові Python використовується динамічна типізація: типи даних визначаються автоматично, змінну не потрібно оголошувати.

У таблиці 1.1 представлені основні вбудовані типи об’єктів мови програмування Python.

Таблиця 1.1 – Основні вбудовані об’єкти в мові програмування Python

Тип	**Приклад**
Число	15, -103, 14.28
Рядок	'example', s2="it's"
Перелік	[1, "abc"]
Словники	{'rec1': 10, 'rec2': 22}

Продовження таблиці 1.1

Тип	**Приклад**
Кортежі	(-23, 15, "text")
Множини	set(‘str’), {‘s’, ‘t’, ‘r’}
Додаткові типи	Функції, файли, модулі, класи

Таблиця 1.1 містить далеко не повний перелік об’єктів, тому що об’єктами є всі дані, оброблення яких виконується в програмах мовою Python.

Приклад програми з використанням об’єктів числових та рядкових типів:

Python

```
percentage = 20.5 / 100
sum_before_taxes = 7052.30
res = sum_before_taxes * percentage
mess = 'calculated:'
print(mess+str(res))
```

У таблиці 1.2 представлені шаблони виклику деяких вбудованих рядкових методів.

Таблиця 1.2 – Основні рядкові методи

Метод	**Опис**
str.capitalize()	Обробляє рядок, перетворюючи його першу літеру на велику, а всі інші залишаючи маленькими
str.count(value, start, end)	Виконує обчислення кількості входжень тексту (підрядка) в рядку
str.endswith(value, start, end)	Виконує перевірку завершення даного рядка заданим набором символів

Продовження таблиці 1.2

Метод	**Опис**
str.find(value, start, end)	Виконує пошук тексту (підрядка) в рядку зліва
str.format(value1, value2...)	Форматування рядка
str.isalnum()	Перевірка того, чи складається рядок повністю з літер або цифр
str.isalpha()	Перевірка того, чи складається рядок повністю з літер
str.islower()	Перевірка того, чи складається рядок повністю з символів у нижньому регістрі
str.isnumeric()	Перевірка того, чи складається рядок повністю з цифр
str.isprintable()	Перевірка того, чи складається рядок повністю з друкованих знаків
str.isspace()	Перевірка того, чи складається рядок повністю з пробільних символів
str.isupper()	Перевірка того, чи складається рядок повністю з символів у верхньому регістрі
str.lstrip(characters)	Обробляє рядок, вилучаючи з нього тільки початкові, розташовані поспіль символи, які є одними з символів з рядка characters
str.partition(sep)	Знаходить входження підрядку в рядок і розділяє за цим входженням рядок на три частини
str.replace(oldvalue, newvalue, count)	Виконує заміну входжень підрядків oldvalue в рядку на рядок newvalue, повторюючи це count разів

Продовження таблиці 1.2

Метод	**Опис**
str.rfind(value, start, end)	Виконує пошук тексту (підрядка) в рядку, починаючи з кінця
str.rpartition(sep)	Знаходить останнє входження підрядку в рядок і розділяє за цим входженням рядок на три частини
str.rstrip(characters)	Обробляє рядок, вилучаючи з нього тільки кінцеві, розташовані поспіль символи, які є одними з символів з рядка characters
str.split(separator, maxsplit)	Виділяє слова з рядку, розділяючи їх за допомогою символів з рядка separator, у кількості maxsplit
str.splitlines(add)	Розділяє текст на рядки, залишаючи символи відділення або ні (визначає параметр add)
str.startswith(value, start, end)	Виконує перевірку початку даного рядка заданим набором символів
str.strip(characters)	Обробляє рядок, вилучаючи з нього початкові, розташовані поспіль символи, які є одними з символів з рядка characters, та аналогічні символи, розташовані в кінці рядка
str.swapcase()	Обробляє рядок, змінюючи регістр всіх символів (верхній на нижній, нижній – на верхній)
str.upper()	Обробляє рядок, збільшуючи регістр всіх символів

У таблиці 1.3 представлені основні операції, які використовуються під час роботи з переліками.

Таблиця 1.3 – Літерали переліків та операції

Операція	**Опис**
l = [20, 7, 1004, 2]	Створення переліку
l = list('text')	Створення переліку за допомогою конструктора
len(l)	Отримання довжини переліку
l[2]	Витягання значення з переліку за індексом
l[1:3]	Отримання зрізу
l+l2	Конкатенація переліків
l.append(52)	Внесення елемента в перелік
l.count(2)	Визначення кількості входжень елемента в перелік
l.extend([9, 10])	Розширення переліку
l.index(20)	Визначення індексу входження елемента в перелік
l.insert(1, 12)	Внесення елемента в перелік на задану позицію (перший аргумент)
l.pop()	Вилучення останнього елемента з переліку
l.remove(1)	Вилучення елемента з переліку за значенням
l.reverse()	Зміна порядку елементів у переліку на протилежний
l.sort()	Сортування переліку
del L[1]	Вилучення елемента з переліку за позицією

У таблиці 1.4 представлені основні операції, які використовуються під час роботи зі словниками.

Таблиця 1.4 – Літерали словників та операції

Операція	**Опис**
v = {'attr1':	Створення словника з парами ключ-значення

Продовження таблиці 1.4

Операція	**Опис**
10, 'attr2': 20}	
v = dict(attr1 = 'Name', attr2=100)	Створення словника за допомогою конструктора
v['attr1']	Витягання значення зі словника за ключем
'attr1' in v	Перевірка існування ключа в словнику
v.copy()	Копіювання словника
v.items()	Отримання переліку пар з ключів і значень зі словника
v.keys()	Отримання переліку ключів зі словника
v.values()	Отримання переліку значень зі словника
v.update(v2)	Об’єднання словників
len(v)	Отримання довжини словника
v[key] = 32	Внесення значення в словник за ключем
del v[key]	Вилучення ключа зі словника

У таблиці 1.5 представлені основні операції, які використовуються під час роботи з кортежами.

Таблиця 1.5 – Літерали кортежів та операції

Операція	**Опис**
t = ('value', 18.3, 19)	Створення кортежу
t = tuple('str')	Створення кортежу за допомогою конструктора
len(t)	Отримання довжини кортежу
t[2]	Витягання значення з кортежу за індексом
t[1:3]	Отримання зрізу

Продовження таблиці 1.5

Операція	**Опис**
t+t2	Конкатенація кортежів
t.count(19)	Визначення кількості входжень елементу в кортеж
t.index(19)	Визначення індексу входження елементу в кортеж

У таблиці 1.6 представлені основні операції, які використовуються під час роботи з файлами.

Таблиця 1.6 – Основні операції над файлами

Операція	**Опис**
fstream = open('data.txt', 'r')	Відкриття файлу для його читання (rb для читання двійкових даних)
data = fstream.read()	Прочитання всього файлу одразу
data = fstream.read(N)	Прочитання N символів (байт) файлу
data = fstream.readline()	Прочитання текстового рядка
ofstream.write(data)	Запис рядка у файл
ofstream.close()	Закриття файлу
ofstream.seek(N)	Зміщення поточної позиції в файлі

У таблиці 1.7 представлені основні інструкції, які використовуються в процесі створення програм мовою Python.

Інструкція	**Приклад**
Присвоювання	a, b = 'text', 10
Виклик функції	fun(data)

break	while True:
	  if not fun():
	  break

import	import lib

Продовження таблиці 1.7

Інструкція	**Приклад**

continue	while True:
	  if fun():
	  continue
	  print(message)

def	def fun(a, *b): return a+b[0]
if (elif-else)	if 't' in info: print(message)
for	for x in onelist: print(x)
from	from lib import cl

global	x = 15
	def fun():
	  global x
	  x = 'value'

pass	def fun(): pass

while	while l:
	  print(message)
	  l.pop()

yield	def fun(n):
	  for i in n:
	  yield i*2

1.2 Використання функціонального програмування для побудови програмної системи

Функціональне програмування – це стиль написання програм через створення набору функцій.

Функціональне програмування є однією з парадигм, які підтримує мова програмування Python як мультипарадигмальна мова програмування.

Поняття функції використовується як в функціональному програмуванні, так і за імперативної парадигми.

Розглянемо приклад функції, написаної не у функціональному стилі.

Python

```
val = 0
def incfun1():
    global val
    val += 1
```

Аналогічна функція, створена в функціональному стилі має наступний вигляд:

Python

```
def incfun2(a):
    return a+1
```

До функцій, які забезпечують реалізацію базових принципів функціонального програмування в мові Python, належать:

– map;
– reduce;
– filter.

1.3 Управління ризиками якості програмного коду

Рефакторинг – підхід до розроблення програмного забезпечення з удосконаленням коду шляхом послідовності семантично коректних перетворень, зміна внутрішньої структури програмного забезпечення для полегшення розуміння коду і здешевлення модифікації [15]-[16].

Підстави для проведення рефакторингу:

– код дублюється;

– метод занадто довгий;

– цикл занадто довгий або рівень вкладеності тіла циклу занадто великий;

– клас має погану зв’язність;

– інтерфейс класу не формує узгоджену абстракцію;

– метод має занадто багато параметрів;

– окремі частини класу змінюються незалежно від інших частин;

– під час зміни програми потрібно паралельно змінювати декілька класів;

– доводиться паралельно змінювати декілька ієрархій наслідування;

– споріднені дані, які використовуються разом, не організовані в клас;

– назва класу чи методу має ім'я, яке недостатньо точно відповідає змісту;

– занадто поширене використання глобальних змінних;

– складний код пояснюється за допомогою коментарів;

– підклас використовує тільки деяку частину методів батьківського класу;

– об’єкт-посередник нічого не виконує;

– дані–члени класу відкриті [15]-[16].

Напрямки проведення рефакторингу можуть включати наступні:

– рефакторинг даних;

– рефакторинг на рівні окремих операторів;

– рефакторинг на рівні окремих методів;

– рефакторинг реалізації класів;

– рефакторинг інтерфейсів класів;

– рефакторинг на рівні системи [15]-[16].

1.4 Порядок виконання розрахунково-графічного завдання

Для виконання розрахунково-графічного завдання необхідно виконати наступні кроки:

1. Отримати у викладача індивідуальне завдання для реалізації програмного забезпечення.

2. Виконати системний аналіз предметної області, представляючи результати у вигляді відповідних діаграм.

3. Визначити принципи і особливості розроблення програмного забезпечення на основі системного підходу.

4. Реалізувати програмне забезпечення у відповідності з принципами, визначеними на кроці 3.

5. Виконати рефакторинг програмного забезпечення.

6. Виконати тестування програмного забезпечення.

7. Оформити пояснювальну записку.

2 РОЗРАХУНКОВО-ГРАФІЧНЕ ЗАВДАННЯ № 2 ЗАТОСУВАННЯ СИСТЕМНОГО ПІДХОДУ ПІД ЧАС РОЗРОБЛЕННЯ КОМПОНЕНТНО-ОРІЄНТОВАНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Розроблення програмних систем на основі компонентів

Розроблення та використання компонентів є основою компонентно-орієнтованого програмування. Дана парадигма програмування реалізується в багатьох технологіях, зокрема в мові програмування Java за допомогою технології JavaBeans, відомі також розповсюджені рішення CORBA, COM, SOAP. Платформа .NET реалізує компонентно-орієнтований підхід, забезпечуючи його використання для підтримуваних мов програмування.

В основі створення компонента .NET лежить клас .NET, наприклад:

C#

```
public class MyClass : IClassInterface
{
   public string GetMessage()
   {
      return "Hello";
   }
}
```

Цей клас створюється на основі організації відповідного інтерфейсу:

C#

```
public interface IClassInterface
{
  string GetMessage();
}
```

Компоненти .NET можуть належати збиранням на основі EXE (збирання застосувань) або DLL (бібліотечні збирання).

Для того щоб створити нове бібліотечне збирання C# у Visual Studio, необхідно скористатися меню File → New → Project... Після того як на екрані з'явиться діалогове вікно New Project, необхідно обрати Visual C# в якості типу проєкту, після чого обрати Windows, а у шаблонах – Class Library. Необхідно також задати розташування проєкту та ввести ім'я (наприклад, MyClassLib).

Після створення проєкту за допомогою меню можна додати новий клас, інтерфейс, компонентний клас тощо. Для цього призначений пункт меню Project → Add Class, який виводить на екран діалогове вікно Add New Item.

Для того щоб використати розроблений компонент у клієнтському застосуванні, необхідно виконати наступні дії:

а) створити новий проєкт Windows Application Form;

б) додати посилання на створене раніше збирання, обравши в меню Project → Add Reference…, після чого на екрані з'явиться діалогове вікно Reference Manager, у якому необхідно обрати створене збирання (обрати варіант Browse та вказати шлях до збирання);

в) додати директиву using для простору імен AssemblyDemo namespace;

г) використати в тексті програми розроблений компонент як звичайний клас, наприклад:

C#

```
using System;
using System.Windows.Forms;
using AssemblyDemo;


  partial class ClientForm : Form
  {
    void OnClicked(object sender,EventArgs e)
    {
      IClassInterface obj = new MyClass( );
      string message = obj.GetMessage( );
```

C#

```
            MessageBox.Show(message);
        }
    }
```

2.2 Порядок виконання розрахунково-графічного завдання

Для виконання розрахунково-графічного завдання необхідно виконати наступні кроки:

1. Отримати у викладача індивідуальне завдання для реалізації компоненто-орієнтованого програмного забезпечення і визначити функціональні вимоги до програмної системи, яка має бути створена в результаті.
2. Виконати системний аналіз предметної області, представляючи результати у вигляді відповідних діаграм.
3. Виконати аналіз ризиків, пов'язаних зі всіма етапами життєвого циклу програмного забезпечення, що розробляється.
4. Виконати оцінювання виявлених ризиків і визначити заходи управління виявленими ризиками. У подальшому виконати моніторинг ризиків, переглядаючи кроки 3-4 під час виконання всіх наступних кроків.
5. Визначити перелік критичних завдань для реалізації проєкту, спираючись на результати попередніх кроків.
6. Виконати проєктування програмної системи у відповідності з індивідуальним завданням на основі створення відповідних діаграм.
7. Розробити програмну систему у відповідності з індивідуальним завданням та результатами попередніх кроків. Усі функції системи повинні бути реалізовані у вигляді необхідної кількості програмних компонентів .NET мовою програмування C#. Клієнтське застосування повинно використовувати функції з компонентів через інтерфейси. Програмна система обов'язково має відповідати вимогам безпеки даних та програмного коду.
8. Виконати тестування розробленої програмної системи.
9. Виконати рефакторинг та оптимізацію коду, обґрунтовуючи прийняті рішення.

10. Проаналізувати ризики, які реалізувались, але не були враховані.

11. Реалізувати кроки 3-4 для подальших етапів життєвого циклу програмного забезпечення, що передбачають запуск і супровід готового продукту.

12. Оцінити результуючий вплив ризиків і проаналізувати прийняті в процесі роботи над програмним забезпеченням рішення відносно ризиків.

13. Оформити пояснювальну записку.

3 ВИМОГИ ДО ЗМІСТУ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Пояснювальна записка до розрахунково-графічного завдання оформлюється у відповідності з вимогами ДСТУ 3008:2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання» [14].

3.1 Вступ

Вступ розташовують на окремій сторінці, у вступі коротко викладають: оцінку сучасного стану проблеми, відмічаючи практично розв'язані задачі, прогалини знань, що існують у даній галузі, провідні фірми та провідних вчених і фахівців даної галузі; світові тенденції розв'язання поставлених задач; актуальність даної роботи та підставу для її виконання; мету роботи та галузь застосування; взаємозв'язок з іншими роботами [14].

3.2 Основна частина

Основна частина пояснювальної записки складається з розділів, підрозділів, пунктів і підпунктів. Основна частина другої роботи містить наступні розділи:

– проєктування системи;

– опис програми;

– управління ризиками програмного забезпечення.

Розділ з проєктування системи повинен включати наступну інформацію:

– вимоги до програмного забезпечення;

– вхідні і вихідні дані;

– спосіб роботи з даними (структура бази даних) з використанням відповідної діаграми;

– прототип графічного інтерфейсу користувача;

– систему об'єктів, які використовуються в предметній області, що аналізується, з описом основних атрибутів і методів даних об'єктів та використанням відповідної діаграми.

– функціональні моделі основних процесів.

Розділ з опису програми повинен містити наступні дані:

– функціональну схему програмного забезпечення;

– структурну схему програмного забезпечення (у вигляді діаграми компонентів);

– опис модулів реалізованого програмного забезпечення з визначенням основних функцій;

– опис роботи з програмним забезпеченням з наведенням усіх необхідних скріншотів;

– опис повідомлень та оброблення критичних ситуацій.

Розділ з управління ризиками програмного забезпечення повинен включати наступну інформацію:

– аналіз ризиків програмного забезпечення за етапами: за кожним етапом розробки програмної системи має бути зазначено, які саме ризики було виявлено, включаючи проблеми та вузькі місця, які виникли в процесі розроблення;

– опис використаних методів управління ризиками з приведенням самого ризику, обґрунтуванням прийнятих рішень та у випадках, де управління здійснювалось заходами на відповідному рівні, наведенням відповідних фрагментів програмного коду.

У підрозділі з аналізу ризиків повинно бути для кожного ризику визначено ймовірність його настання, рівень впливу на розклад проєкту, рівень впливу на витрати проєкту та рівень впливу на досягнення результатів проєкту, а також співставлена зона критичності. Усі ці значення мають бути приведені для стану до застосування управління даним ризиком («до управління», тобто фактично під час аналізу) та після завершення реалізації ризику або заходу з управління («після управління»).

Для оцінювання ймовірності та рівня впливу може використовуватися або числове значення (кількісні ризики), або категорія (якісні ризики). Категорії можуть належати наступному набору: дуже низький, низький, помірний, сильний, дуже сильний.

У випадку використання категорій наступні ситуації можна вважати такими, що відносяться до зони некритичних:

– вплив загалом є дуже низьким;

– вплив є низьким, а ймовірність не вище помірної;

– ймовірність є дуже низькою, а вплив не є дуже сильним.

Наступні ситуації, не включаючи описані некритичні, потребують додаткової уваги і належать до зони важливих:

– за сильної (високої) або дуже сильної ймовірності вплив є низьким або помірним;

– за помірної або низької ймовірності вплив є помірним або високим;

– за дуже низької ймовірності вплив є дуже сильним.

Наступні ситуації, не включаючи описані вище, належать до зони критичних:

– ймовірність настання і одночасно вплив є дуже сильними (високими) або сильними (високими);

– вплив є дуже сильним (високим), а ймовірність настання помірною або низькою.

Для кількісних ризиків може використовуватися наступна система перетворення значення ймовірності у відповідні категорії:

– не більше 10 % для дуже низької;

– більше 70 % для дуже сильної (високої);

– більше 50 % для сильної;

– більше 30 % для помірної;

– більше 10 % для низької.

Переведення конкретних значень впливу кількісного ризику в категорії для подальшого визначення зони критичності залежить від відповідних запланованих показників проєкту (витрат, тривалості, результатів).

3.3 Висновки

Висновки вміщують безпосередньо після викладення суті записки, починаючи з нової сторінки: у висновках наводять найбільш важливі результати, одержані в розрахунково-графічному завданні, виконують оцінку одержаних результатів роботи з урахуванням світових тенденцій вирішення поставленої задачі; можливі галузі використання результатів роботи; наукову, соціальну значущість роботи, у висновках необхідно наголосити на якісних та кількісних показниках отриманих результатів, обґрунтувати достовірність результатів, викласти рекомендації щодо їх використання [14].

Висновки повинні займати не менше однієї повної сторінки.

3.4 Правила оформлення пояснювальної записки

Відповідно до ДСТУ 3008:2015 та діючих норм встановлено наступні основні правила оформлення пояснювальної записки:

– основний текст документу набирається з використанням шрифту Times New Roman розміром 14 пунктів із міжрядковим інтервалом 1,5, розмір абзацу – 1,5 см;

– розмір полів документа: верхнє і нижнє – не менше 20 мм, ліве – не менше 25 мм, праве – не менше 10 мм;

– усі сторінки мають наскрізну нумерацію, починаючи з титульного аркуша, номер сторінки друкується в правому верхньому куті, номер сторінки не друкується на титульному аркуші;

– текст пояснювальної записки має бути структурований на розділи та підрозділи, які повинні мати заголовки, пункти і підпункти також можуть мати заголовки;

– відстань між заголовком і текстом, а також між текстом і наступним заголовком повинна бути не менше, ніж два рядки;

– міжрядковий інтервал у тексті заголовку, а також між двома заголовками, повинен бути таким, як у тексті, тобто 1,5 інтервали;

– заголовки розділів виконуються прописними літерами з форматуванням за центром розміром 14 пунктів жирним шрифтом;

– підзаголовки першого рівня виконуються строковими літерами розміром 14 пунктів з абзацу, як правило, жирним шрифтом;

– підзаголовки другого та інших рівнів – з абзацу розміром 14 пунктів, як правило, жирним шрифтом;

– посилання на друковані джерела мають подаватися в квадратних дужках: бібліографічні описи в переліку посилань наводять за порядком першого згадування в тексті;

– таблицю слід розташовувати безпосередньо після тексту, в якому вона згадується вперше або на наступній сторінці: на таблицю мають бути посилання в тексті, таблиці нумерують наскрізно арабськими цифрами, крім таблиць у додатках, дозволено таблиці нумерувати в межах розділу (номер таблиці складається з номера розділу та порядкового номера таблиці, відокремлених крапкою, наприклад, «Таблиця 2.1» – перша таблиця другого розділу), назву таблиці друкують з великої літери і розміщують над таблицею з абзацного відступу;

– якщо таблиця не може бути розміщена на одній сторінці, то її продовження переносять на наступну сторінку: у разі поділу таблиці на частини дозволено її головку чи боковик заміняти відповідно номерами колонок або рядків, нумеруючи їх арабськими цифрами в першій частині таблиці, слово «Таблиця» подають лише один раз над першою частиною таблиці, над іншими частинами таблиці з абзацного відступу друкують «Продовження таблиці (номер)» або «Кінець таблиці (номер)» без повторення її назви;

– заголовки колонок таблиці починають з великої літери, підзаголовки – з малої, якщо вони становлять 1 речення з заголовком;

– рисунок подають одразу після тексту, де вперше посилаються на нього, або якнайближче до нього на наступній сторінці;

– рисунки нумерують наскрізно арабськими цифрами, крім рисунків у додатках: дозволено рисунки нумерувати в межах кожного розділу, у цьому разі номер рисунка складається з номера розділу та порядкового номера рисунка в цьому розділі, які відокремлюють крапкою, наприклад, «Рисунок 3.2» – другий рисунок третього розділу (назву рисунка друкують з великої літери та розміщують під ним посередині рядка, наприклад, «Рисунок 2.1 – Схема устаткування»);

– за необхідністю можуть бути використані переліки: перед переліком ставлять двокрапку, якщо подають переліки одного рівня підпорядкованості, на які немає посилань, то перед кожним із переліків ставлять знак «тире», якщо у тексті є посилання на переліки, підпорядкованість позначають малими літерами української абетки, далі – арабськими цифрами, далі – через знаки «тире», після цифри або літери певної позиції переліку ставлять круглу дужку, текст кожної позиції переліку треба починати з малої літери з абзацного відступу відносно попереднього рівня підпорядкованості [14].

У одному з додатків обов’язково має бути наведено текст програми.

Кожен додаток повинен починатися з нової сторінки, додаток повинен мати заголовок, надрукований вгорі малими літерами з першої великої симетрично відносно тексту сторінки, посередині рядка над заголовком малими літерами з першої великої повинно бути надруковано слово "Додаток ___" і велика літера, що позначає додаток [14].

ЛІТЕРАТУРА

1. Навчальний посібник з дисципліни «Системний аналіз» для здобувачів спеціальності 122 – комп'ютерні науки [Текст] / В.М. Тонконогий, В.О. Вайсман, Л.В. Бовнегра, К.Г. Кіркопуло. – Одеса: Нац. ун-т «Одеська політехніка», 2022. – 84 с.

2. Панкратова, Н.Д. Системний аналіз [Текст] : теорія та застосування: підручник / Н.Д. Панкратова. – К.: Вид-во «Наукова думка» НАН України, 2019. – 352 с

3. Прокопенко, Т.О. Теорія систем і системний аналіз [Електронний ресурс] : навчальний посібник / Т.О. Прокопенко. – Черкаси: ЧДТУ, 2019. – 139 с.

4. Lutz, M. Learning Python [Текст] / M. Lutz ; 5th Edition. – O'Reilly Media, 2013. – 1643 p.

5. Коноваленко, І.В. Платформа .NET та мова програмування C# 8.0 [Текст]: навчальний посібник / І.В. Коноваленко, П.О. Марущак. – Тернопіль: ФОП Паляниця В. А., 2020. – 320 с.

6. Васильєв, О. Програмування мовою Python [Текст] / О. Васильєв. – Тернопіль: "Навчальна книга – Богдан", 2019. – 504 с.

7. Висоцька, В.А. Python [Текст] : алгоритмізація та програмування / В.А. Висоцька, О.В. Оборська. – Львів: Видавництво «Новий Світ – 2000», 2021. – 514 с.

8. Заяць, В.М. Логічне і функціональне програмування. Системний підхід. Підручник. – 2-ге видання, випр. та доповн. [Текст] / В.М. Заяць, М.М. Заяць. – Рівне: НУВГП, 2018. – 422 с.

9. Шушура, О.М. Системний аналіз [Текст] : навчальний посібник / О.М. Шушура, Н.К. Шатохіна. – К.: Редакційно-видавничий центр Державного університету телекомунікацій, 2019. – 63 с.

10. Python 3.11.3 documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/

11. Pygame Front Page – pygame v2.4.0 documentation [Електронний ресурс]. – Режим доступу: https://www.pygame.org/docs/

12. C# docs – get started, tutorials, reference. Microsoft Learn [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/uk-ua/dotnet/csharp/

13. Vu, H. Component Oriented Programming [Текст] : Object-Oriented and Beyond / Hieu Vu. – LAP LAMBERT Academic Publishing, 2017. – 188 p.

14. ДСТУ 3008:2015 Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання [Текст]. – К. : ДП «УкрНДНЦ», 2016. – 26 с.

15. Fowler, M. Refactoring [Текст] : Improving the Design of Existing Code / M. Fowler. – Addison-Wesley Professional, 2018. – 448 p.

16. Mcconnell, S. Code Complete [Текст] : A Practical Handbook of Software / S. Mcconnell. – Microsoft Press US, 2004. – 960 p.