ДЕРЖАВНА СЛУЖБА УКРАЇНИ З НАДЗВИЧАЙНИХ СИТУАЦІЙ

ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ БЕЗПЕКИ ЖИТТЄДІЯЛЬНОСТІ

Ольга Смотр, Олександр Придатко, Ігор Малець

ОСНОВИ ПРОГРАМУВАННЯ

(PYTHON, JAVA)

Лабораторний практикум

для бакалаврів спеціальності 122 «Комп‘ютерні науки»

Львів-2019

Основи програмування (PYTHON, JAVA): лабораторний практикум для студентів спеціальності 122 «Комп‘ютерні науки»; денної та заочної форм навчання / уклад.: О.О. Смотр., О.В. Придатко, І.О. Малець. – Львів, 2019. – 134 с.

Рецензенти:

Луб П.М., канд. техн. наук, доцент, доцент кафедри інформаційних систем та технологій Львівського національного аграрного університету;

Кухарська Н.П., канд. фіз.-мат. наук, доцент, доцент кафедри управління інформаційною безпекою Львівського державного університету безпеки життєдіяльності.

Укладачі: к.т.н. О.О. Смотр, к.т.н. О.В. Придатко, к.т.н. І.О. Малець.

Рекомендовано до друку рішенням вченої ради

Львівського державного університету безпеки життєдіяльності

(протокол № 3 від 30 жовтня 2019 року)

ЗМІСТ

ВСТУП

Сучасні тенденції широкого використання інформаційних технологій та засобів комп'ютерного моделювання у всіх галузях економіки висувають нові вимоги до рівня підготовки студентів вищих технічних закладів у галузі програмування та ІТ Створення достатньої теоретичної та практичної бази для студентів з цих предметів дозволяє їм швидко вирішити поточні проблеми, приділяючи мінімальний час на переробку. Особливу увагу слід приділити підготовці майбутніх фахівців з розробки сучасних систем.

Запропоноване видання дозволяє студентам без будь-яких базових знань з програмування за час навчання освоїти основні конструкції Python та JAVA (на основі стандартних бібліотек) і впевнено оволодіти навичками алгоритмізації. Це видання створено на основі комп‘ютерних практикумів з дисципліни «Основи програмування», які виконуються у Львівському державному університеті безпеки життєдіяльності для студентів напряму підготовки 122 «Комп‘ютерні науки». Виклад матеріалу дозволяє його використати не тільки викладачам, а й студентам під час самопідготовки. Всі роботи супроводжуються прикладами розв'язання задач. Програми виконувалися в середовищах Python 3.X. та інтегрованому середовищі розробки Eclipse відповідної версію JDK,

Використовуючи доступне програмне забезпечення на власних персональних комп’ютерах і необхідне методичне забезпечення у вигляді розроблених лабораторних практикумів, студенти мають можливість проводити моделювання та виконання лабораторних робіт самостійно та дистанційно. Сукупність набутих навичок дозволяє студентам вже на першому курсі розробляти досить складні алгоритми.

Лабораторний практикум є збіркою завдань та інструкцій для закріплення теоретичних знань, отриманих на лекційних заняттях з дисципліни "Основи програмування".

Кожна лабораторна робота практикуму містить тему, мету, короткі теоретичні відомості, завдання, порядок виконання завдань і список питань для самоперевірки. На захист виконаної студентом лабораторної роботи

оформлюється окремий звіт та представляється електронний документ, із виконаним завданням.

Звіт повинен включати наступні пункти:

- Мета лабораторної роботи.
- Короткі теоретичні відомості.
- Індивідуальне завдання (згідно варіанту) лабораторної роботи.
- Основні етапи виконання роботи та отримані результати.
- Висновки.

Розділ І. *PYTHON*

Практичне заняття № 1.
Ознайомлення з віртуальним навчальним середовищем «Віртуальний університет»

Мета: реєстрація учасників групи в віртуальному навчальному середовищі та здобуття базових навичок щодо користування електронним ресурсом.

Після практичного заняття студенти (курсанти) повинні:

вміти: здійснювати пошук необхідної інформації в віртуальному навчальному середовищі, користуватись методичними матеріалами, виконувати індивідуальні завдання із завантаженням відповідних звітів та проходити індивідуальні тестові завдання.

знати: особливості роботи з віртуальним навчальним середовищем «Віртуальний університет».

План заняття:

1. Основи роботи з віртуальним навчальним середовищем «Віртуальний університет».

2. Перший застосунок в віртуальному навчальному середовищі – складання тестових завдань за темою попередньої лекції.

Завдання

1. Зареєструватись на курс «Основи програмування» у віртуальному навчальному середовищі «Віртуальний університет»

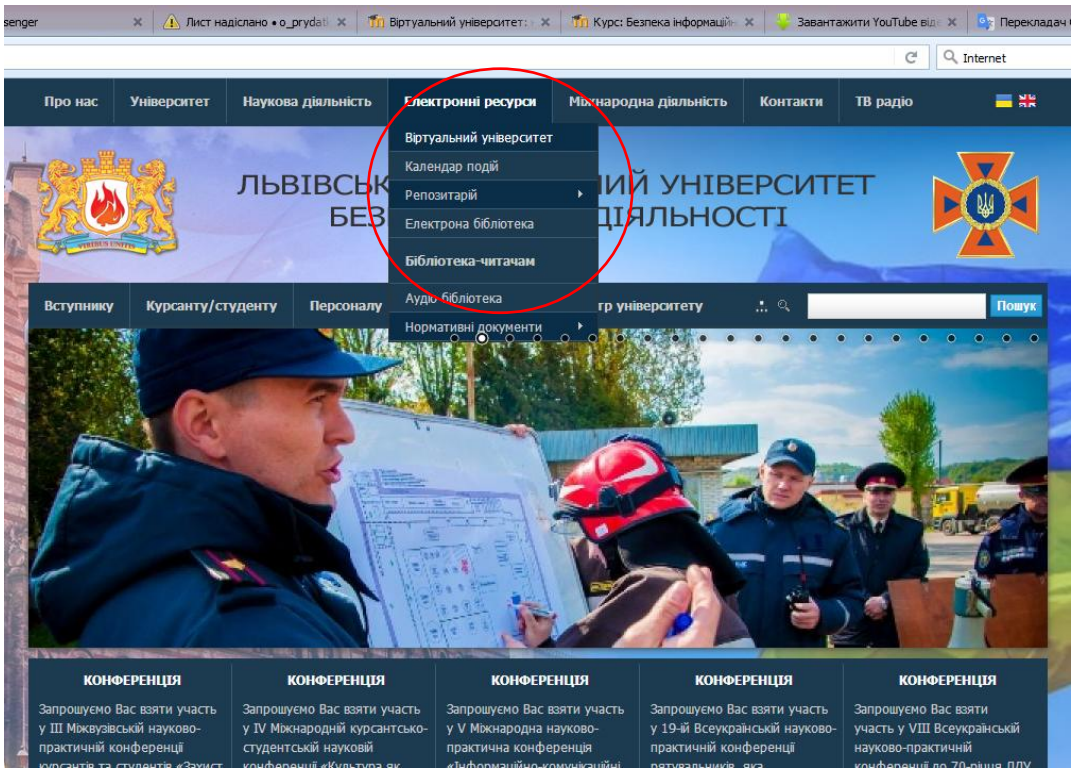
2. Пройти тестові завдання за темою: «Вступ до курсу. Апаратні за програмні засоби ЕОМ».

Аудиторна робота

1. Основи роботи в віртуальному навчальному середовищі «Віртуальний університет»

Доступ до віртуального навчального середовища можна здійснити за адресою http://virt.ldubgd.edu.ua/ або через головну сторінку сайту Львівського

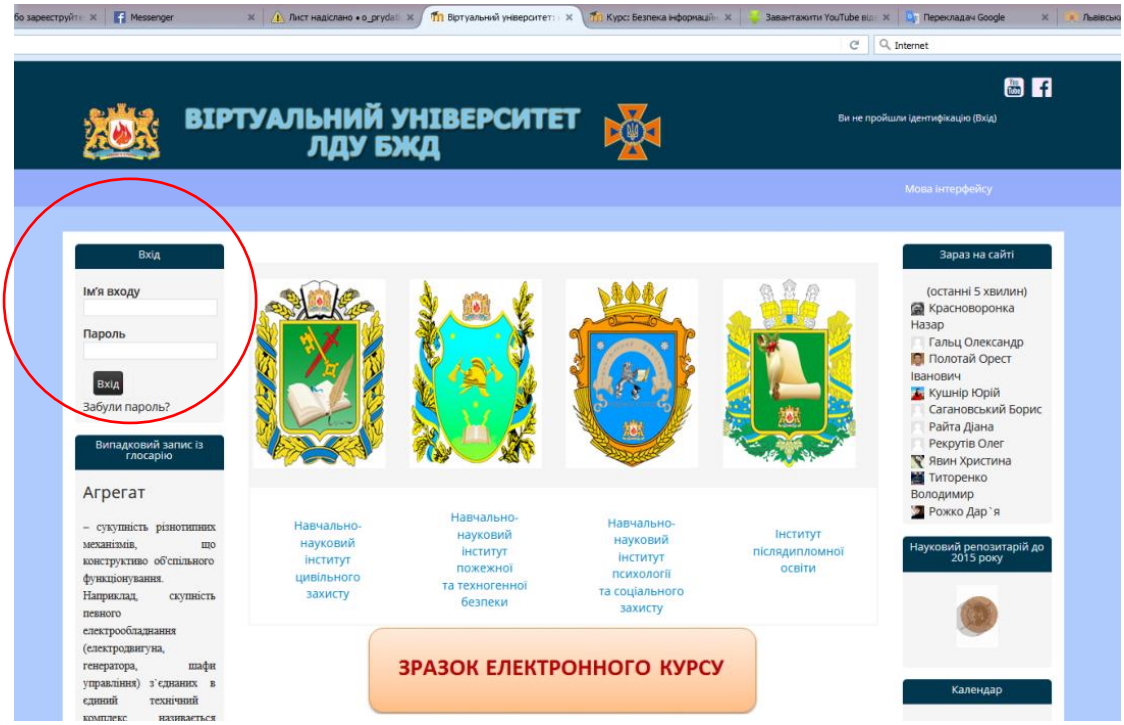
державного університету безпеки життєдіяльності в категорії «електронні ресурси».

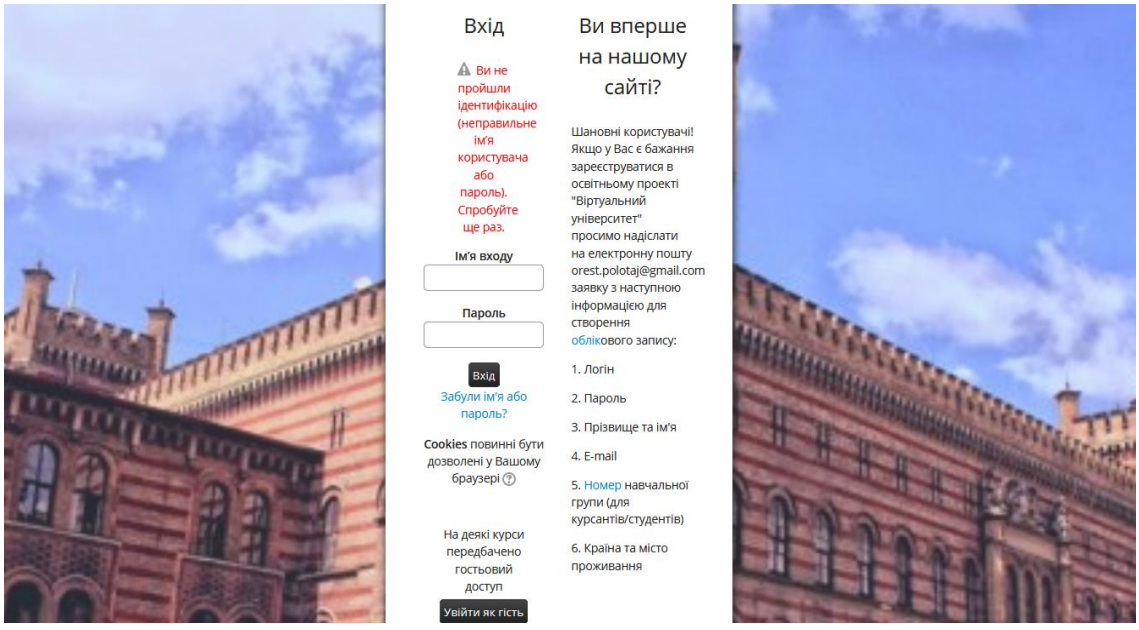


На головній сторінці віртуального навчального середовища розміщено поле «вхід» з відповідними полями «логін» та «пароль». Користувачі, які працюють в середовищі вперше зобов'язані здійснити реєстрацію (створити власний акаунт) з метою подальшого доступу до усіх методичних матеріалів курсу.

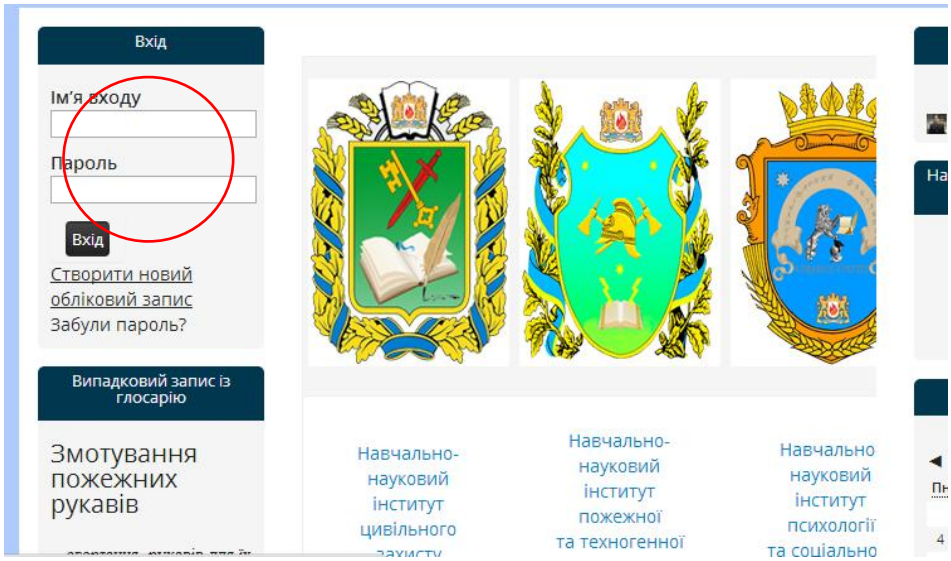
Реєстрація в різних режимах функціонування навчального середовища можлива різним шляхом. За умови закритого доступу реєстрації, в зазначеній області (вхід) буде відсутнє посилання «створити новий обліковий запис». В такому випадку необхідно здійснити «вхід» без введення будь-якого логіну та паролю.

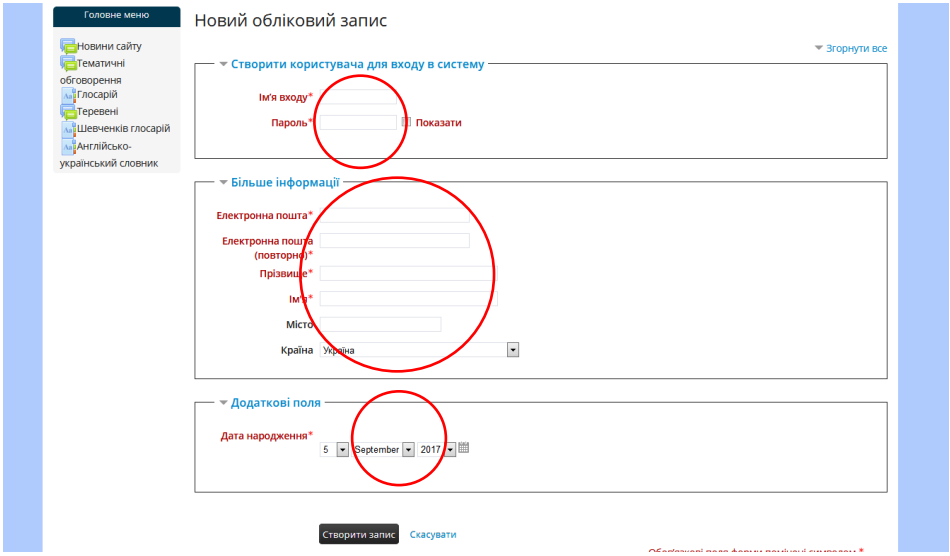
В результаті цієї операції на екрані з'явиться повідомлення з інформацією щодо подальших дій для створення облікового запису (надіслати на електронну пошту orest.polotaj@gmail.com наступну інформацію: логін, пароль, прізвище та ім'я, e-mail, номер навчальної групи, країна та місто проживання).





За умови відкритого доступу до реєстрації, в області «вхід» буде доступне посилання «створити новий обліковий запис», після проходженням за яким користувачеві надається можливість здійснити самостійну реєстрацію з відповідним введення основної інформації.

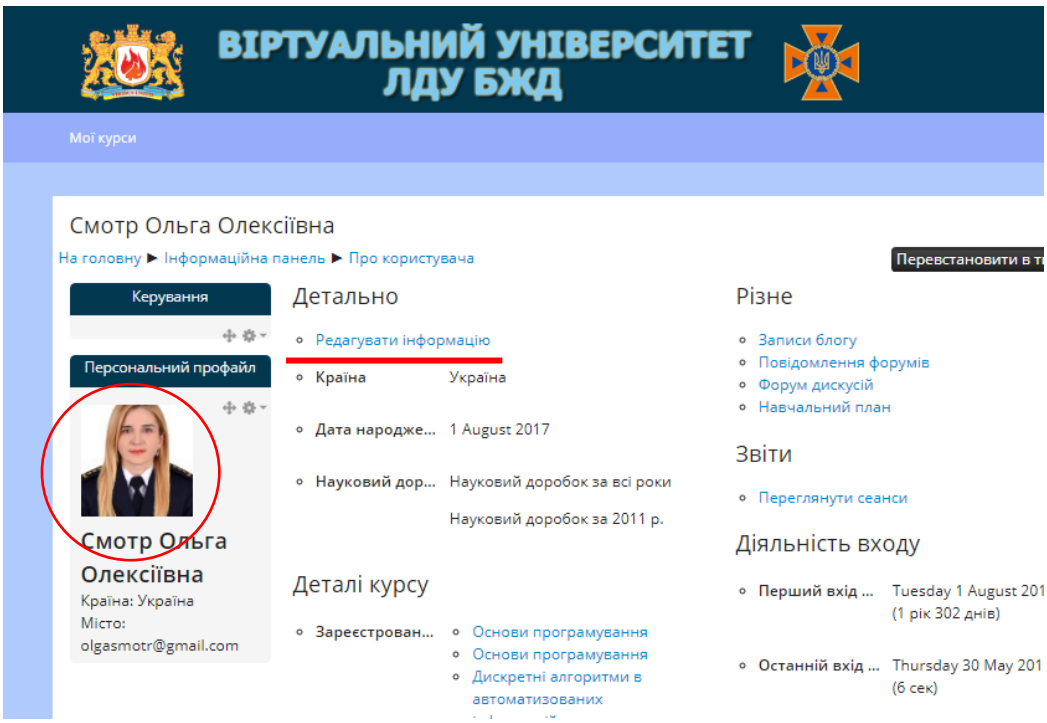




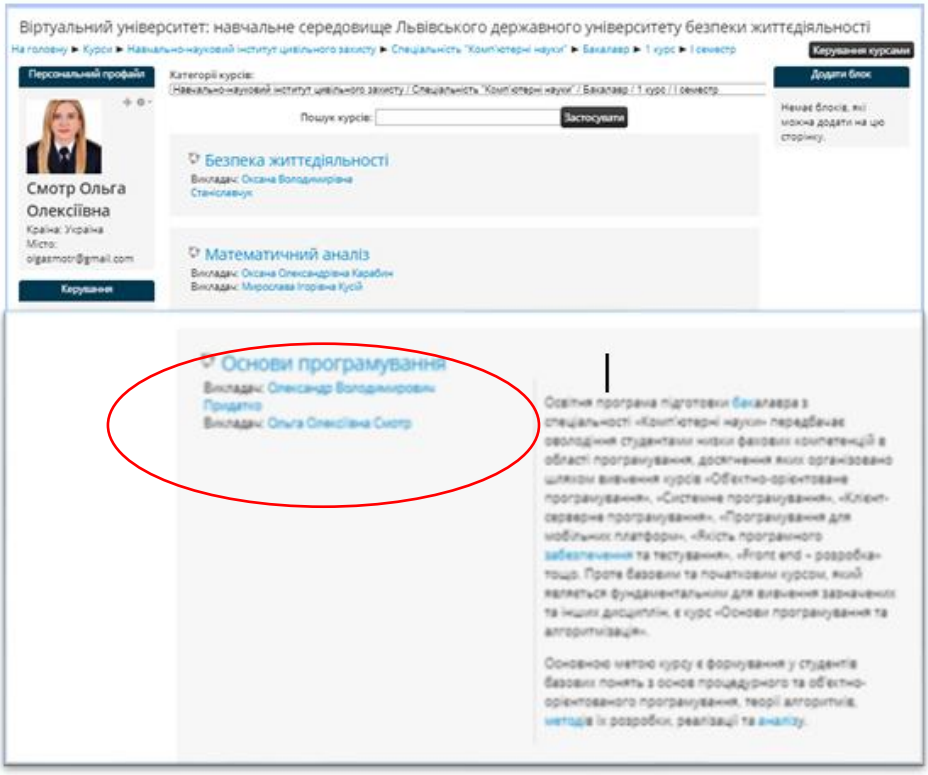
Слід зазначити, що ім'я користувача (логін) та пароль для входу в систему будуть незмінним протягом усього періоду навчання в університеті (за винятком створення нового облікового запису), тому слід віднестись відповідально до їх вибору.

Після введення усіх даних та створення облікового запису, користувачеві за вказаною ним електронною адресою буде надіслано лист із посиланням для підтвердження реєстрації. Рекомендовано вказувати адресу електронної скриньки, яка розміщена на серверах ukr.net, i.ua, або gmail.com. Електронні скриньки «російський» серверів системою не підтримуватимуться.

Після успішної реєстрації, користувачеві, за умови введення логіну та паролю, надається доступ до віртуального середовища. Після входу в віртуальне середовище, замість області «вхід» з'являється персональний профайл з відображенням основної інформації про користувача. Для редагування цієї інформації, а також долучення особистого фото, або будь-якої іншої додаткової інформації, користувачеві необхідно натиснути на область розташування фото особистого профілю та пройти за посиланням «редагувати інформацію».



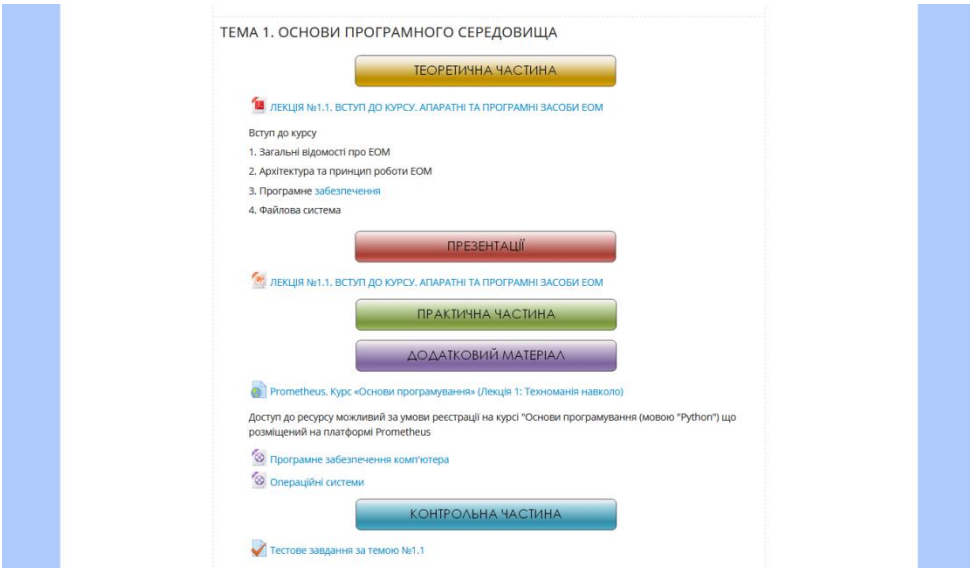
З головної сторінки віртуального середовища надається доступ до будь-якого курсу, який цікавить студента. З метою доступу до курсу «Основи програмування» з головної сторінки віртуального середовища необхідно пройти за посиланням «Навчально-науковий інститут цивільного захисту», після чого обрати спеціальність «Комп'ютерні науки», освітній рівень «бакалавр», курс «1-й», семестр I.



Основи програмування

Для вільного користування методичними матеріалами необхідно здійснити реєстрацію свого облікового запису на курсі. Можливі різні методи реєстрації: самореєстрація, ручне зарахування тощо. Відповідно до режиму реєстрації подальші дії щодо зарахування студента на відповідний електронний курс необхідно узгоджувати з викладачем (адміністратором) курсу.

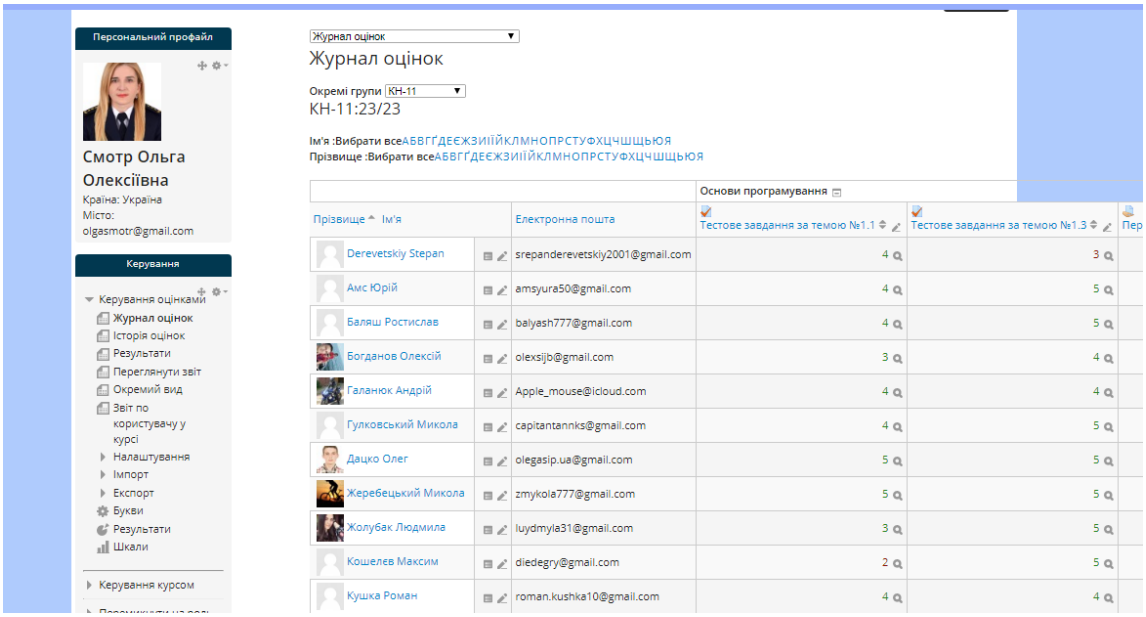
Реєстрація на курсі «Основи програмування» надає можливість користувачеві здійснювати доступ до теоретичного матеріалу, презентацій, практичних завдань, додаткових джерел для поглиблення знань, проходити тестові завдання та завантажувати звіти виконання практичних робіт. Крім того на сторінці курсу міститься інформація щодо основних та додаткових джерел літератури з відповідними посиланнями, а також інтернет-ресурсів, які можуть бути корисними в процесі вивчення дисципліни.



В якості додаткових засобів поглиблення знань на сторінці курсу розміщено глосарій основних термінів, а також додано можливість висловлення власної думки про інформативність та корисність курсу (анонімно).

Усі звіти та основні результати виконання індивідуальних практичних завдань також завантажуються з головної сторінки курсу, після чого рецензуються та оцінюються викладачем із виставленням відповідної оцінки в електронному журналі.

Щодо електронного журналу, то в ньому відображатимуться усі поточні навчальні досягнення студентів із можливістю вільного доступу (без редагування). В журналі відображаються результати складання обов’язкових тестових завдань та виконання індивідуальних практичних завдань.



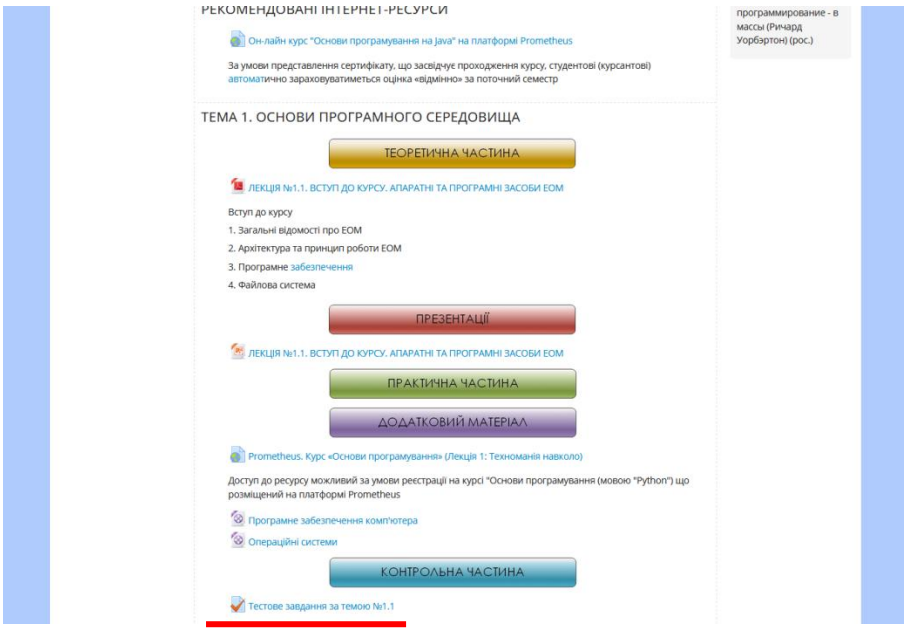
2. Перший застосунок в віртуальному навчальному середовищі – складання тестових завдань за темою попередньої лекції

В якості першого застосунку в віртуальному навчальному середовищі запропоновано проходження тестових завдань за темою попередньої лекції «Вступ до курсу. Апаратні за програмні засоби ЕОМ». Користувачеві надається можливість пройти запропонований тест двічі. Зараховуватиметься результат кращої спроби. Час проходження тестових завдань лімітовано (1 питання – 1 хвилина).

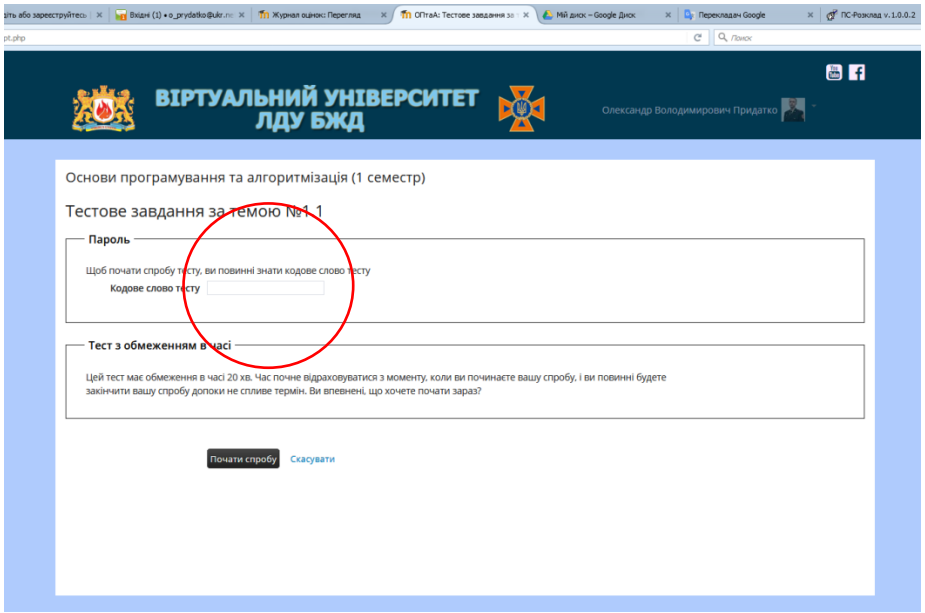
Результат кращої спроби проходження тестування автоматично заноситься до електронного журналу успішності. Перегляд журналу оцінок доступний усім користувачам, які зареєстровані на відповідний курс.

Слід зазначити, що проходження тестових завдань відбуватиметься на початку кожного практичного заняття протягом вивчення курсу. Мета проведення тестування – це визначення рівня засвоєння теоретичного (лекційного) матеріалу. Рівень засвоєння практичних навиків визначатиметься за результатами індивідуальних практичних завдань.

З метою проходження тестових завдань необхідно перейти до категорії «практична частина» відповідної теми та обрати тест за номером лекції.

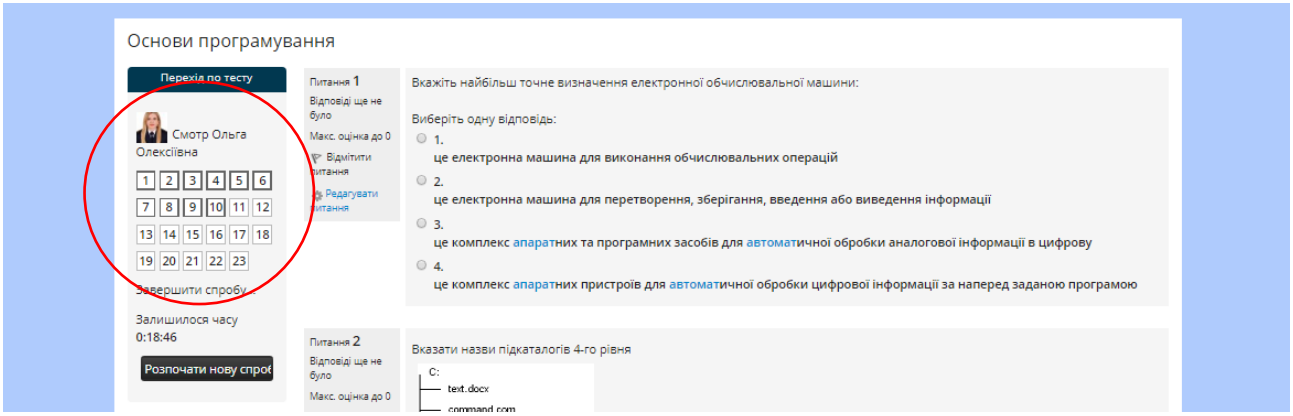


Для доступу до тестових завдань необхідно ввести кодове слово. Кодові слова до окремих тестових завдань є унікальними та будуть доводитись викладачем.



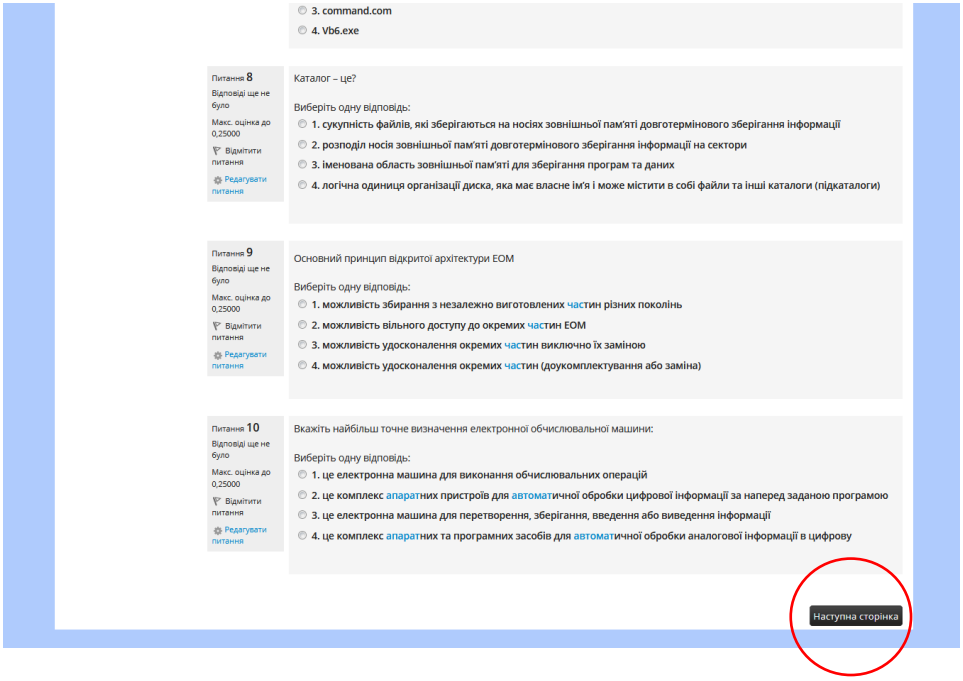
Лічильник обліку часу тестування відображено в лівому верхньому куті сторінки. В зазначеному куті також розміщено інформацію щодо кількості тестових запитань, кількості їх відображення на поточні сторінці тощо.

Найпростіший формат тесту вимагає від користувача вибрати одну правильну відповідь на запитання. В першому застосунку (першому тесті) зустрічатимуться виключно такі завдання. Проте в подальших тестах з курсу зустрічатимуться питання з декількома правильними відповідями, питання з чіткою відповіддю, питання з логічним розв’язком тощо.

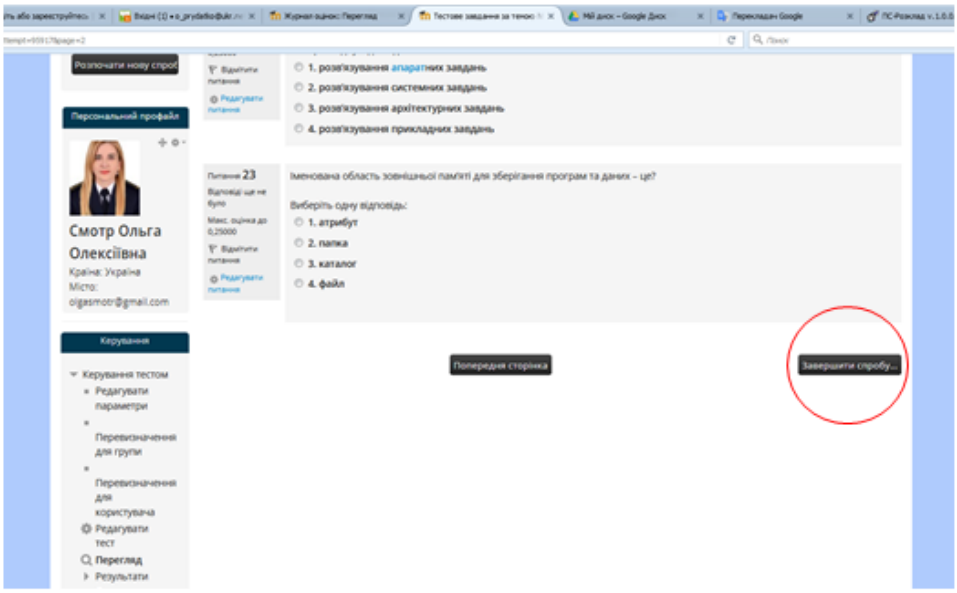


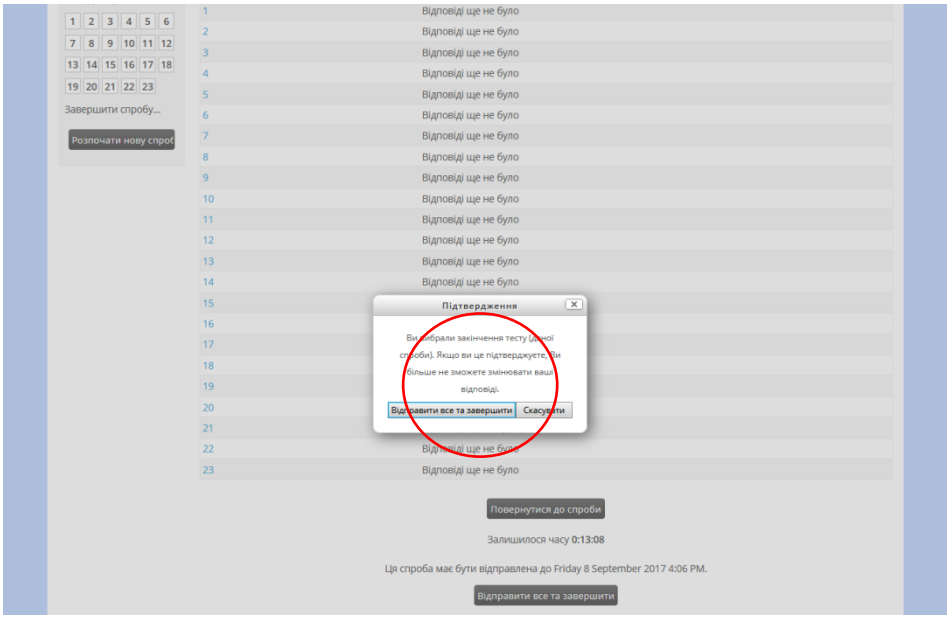
Під час проходження тестування з метою переходу на наступну сторінку тестового завдання із збереженням існуючих відповідей необхідно натиснути на кнопку «наступна сторінка».

Після проходження усього тесту користувачеві необхідно натиснути на кнопку «завершити спробу». Після чого діалогове вікно запропонує відправити результати та завершити спробу.



Важливо! Час проходження тестування лімітовано. У випадку завершення виділеного часу усі результати незавершеної спроби анулюються. Це пов’язано з тим, що система фіксує одночасно всі варіанти відповідей тільки після завершення тестування.





Результати проходження тесту відображаються в верхній частині сторінки, а також, як вже було зазначено раніше, автоматично зараховуються до електронного журналу оцінок.

Максимальна оцінка за тестове завдання складає «5» (відмінно), мінімальна «0» (незадовільно). Для отримання максимальної оцінки студентові необхідно дати правильні відповіді на 90% запитань.

Тест вважається пройденим (зарахованим) за умови одержання позитивної оцінки «3» (задовільно) та вище.

Час між спробами проходження тестування не обмежений. Відповідно користувач може проаналізувати допущені ним помилки під час першої спроби, усунути прогалини в знаннях та повторно покращити свої результати.

Практичне заняття № 2
«Переведення чисел в різні системи числення. Арифметичні операції в системах числення»

Мета: оволодіння базисними поняттями щодо процесів кодування алфавітно-цифрової інформації в ЕОМ.

Після практичного заняття студенти (курсанти) повинні:

вміти: здійснювати переведення чисел між різними системами числення, а також проводити основні арифметичні дії в системах числення..

знати: базові принципи кодування алфавітно-цифрової інформації в ЕОМ.

План:

1. Переведення чисел між різними системами числення.
2. Арифметичні операції в системах числення.
3. Індивідуальне практичне завдання.

Аудиторна робота

1. Переведення чисел між різними системами числення.

З метою закріплення теоретичного матеріалу, який був прослуханий на лекції, розглянемо приклади усіх можливих перетворювань до різних систем числення для десяткового цілого числа **263**.

Спочатку проведемо перетворення числа з 263_{10} в 263_2:

263 : 2 = 131 **(1)**

131 : 2 = 65 **(1)**

65 : 2 = 32 **(1)**

32 : 2 = 16 **(0)**

16 : 2 = 8 **(0)**

8 : 2 = 4 **(0)**

4 : 2 = 2 **(0)**

2 : 2 = **1 (0)**

Результат: $\boldsymbol{263_{10} = 100000111_2}$.

Наступним кроком здійснимо зворотне перетворення з 263_2 в 263_{10}:

$$100000111_2 = 1 \cdot 2^8 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 256 + 4 + 2 + 1 = 263_{10}$$

Результат: ***$100000111_2 = 263_{10}$***;

Далі число 263 перетворимо з 263_{10} в 263_8:

263 : 8 = 32 **(7)**

32 : 8 = **4 (0)**

Результат: ***$263_{10} = 407_8$***;

Наступним кроком з 263_8 в 263_{10}:

$$407_8 = 4 \cdot 8^2 + 7 \cdot 8^0 = 256 + 7 = 263_{10}$$

Результат: ***$407_8 = 263_{10}$***;

Далі проведемо перетворення з 263_{10} в 263_{16}:

263 : 16 = 16 **(7)**

16 : 16 = **1 (0)**

Результат: ***$263_{10} = 107_{16}$***;

Зворотнє перетворення з 263_{16} в 263_{10}:

$$107_{16} = 1 \cdot 16^2 + 7 \cdot 16^0 = 256 + 7 = 263_{10}$$

Результат: ***$107_{16} = 263_{10}$***;

Далі між «машинними» системами числення, а саме з 263_{16} в 263_2 (з використанням таблиці):

7 – 0111

0 – 0000

1 – 0001

Результат: $\boldsymbol{107_{16} = 1\ 0000\ 0111_2}$;

Наступним кроком є зворотне перетворення з 263_2 в 263_{16} (з використанням таблиці) :

0001 0000 0111

0111 – 7

0000 – 0

0001 – 1

Результат: $\boldsymbol{100000111_2}$ = 0001 0000 0111 = $\boldsymbol{107_{16}}$;

Далі з 263_8 в 263_2 (з використанням таблиці):

7 – 111

0 – 000

4 – 100

Результат: $\boldsymbol{407_8}$ = 100 000 111 = $\boldsymbol{100000111_2}$;

І останнє з 263_2 в 263_8 (з використанням таблиці):

100 000 111

111 – 7

000 – 0

100 – 4

Результат: $\boldsymbol{100000111_2}$ = 100 000 111 = $\boldsymbol{407_8}$.

2. Арифметичні операції в системах числення

Перш ніж здійснювати будь-які арифметичні операції в системах числення, розглянемо правила виконання арифметичних дій на прикладі двійкової системи числення.

Таблиця 2.1 – Основні правила виконання арифметичний дій над двійковими числами

Додавання	Віднімання	Множення
0+0=0	0–0=0	0×0=0
1+0=1	1–0=1	1×0=0
0+1=1	1–1=0	0×1=0
1+1=10	0–1=1`1	1×1=1

Слід зауважити, що в таблиці 2.1 не представлено правил виконання операції ділення! Це пов’язано з тим, що ділення в системах числення, зокрема двійковій, виконується на основі почергового множення та віднімання. Відповідно знань правил виконання цих арифметичних операцій (табл. 2.1) буде достатньо.

Також цікавим є той факт, що при додаванні «1» та «1» отримано «10». Число «10» не є десятковим, це окремі значення «1» та «0».

Перевагою двійкової системи числення над десятковою є менша кількість правил виконання арифметичний дій над числами, що власне представлено в таблиці.

Далі розглянемо приклади виконання арифметичних операцій.

Додавання:

$$\begin{array}{llll} \cdot\cdot\ \cdot\cdot & \cdot\cdot\ \cdot & \cdot\ \cdot & \cdot\cdot \\ +\ 1100101_2 & +\ 250361_8 & +\ F2A_{16} & +\ 30A67_{16} \\ \underline{\ \ 1101110_2} & \underline{\ \ 2541714_8} & \underline{\ \ 3BC_{16}} & \underline{\ \ 2AF824_{16}} \\ 11010011_2 & 3012275_8 & 12E6_{16} & 2E028B_{16} \end{array}$$

Віднімання:

$$
\begin{array}{cccc}
\begin{array}{r} \cdot\cdot \\ \_100_2 \\ \underline{\quad 1_2} \\ \_11_2 \end{array} &
\begin{array}{r} \cdot\cdot \\ \_100_8 \\ \underline{\quad 1_8} \\ \_77_8 \end{array} &
\begin{array}{r} \cdot\cdot \\ \_100_{16} \\ \underline{\quad 1_{16}} \\ \_FF_{16} \end{array} &
\begin{array}{r} \cdot \\ \_F05_{16} \\ \underline{\quad D1_{16}} \\ \_E34_{16} \end{array} \\
\begin{array}{r} \cdot\cdot\cdot\ \cdot \\ \_1100101_2 \\ \underline{\quad 101010_2} \\ \_111011_2 \end{array} &
\begin{array}{r} \cdot\cdot \\ \_5407024_8 \\ \underline{\quad 640052_8} \\ \_4546752_8 \end{array} &
\begin{array}{r} \cdot\ \cdot\ \cdot \\ \_2AF8D4_{16} \\ \underline{\quad F0A67_{16}} \\ 1BEE6D_{16} \end{array} &
\begin{array}{r} \cdot\ \cdot\ \cdot \\ \_CF05_{16} \\ \underline{\quad F06_{16}} \\ BFFF_{16} \end{array}
\end{array}
$$

Множення:

$$
\begin{array}{cccc}
\begin{array}{r} \times 1001011_2 \\ \underline{\quad 1001_2} \\ +\ 1001011_2 \\ 1001011_2 \\ \hline 1010100011_2 \end{array} &
\begin{array}{r} \times 47_8 \\ \underline{\quad 23_8} \\ +\ 165_8 \\ 114_8 \\ \hline 1345_8 \end{array} &
\begin{array}{r} +\ 41_{16} \\ \underline{\quad AF_{16}} \\ +3CF_{16} \\ 28A_{16} \\ \hline 2C6F_{16} \end{array} &
\begin{array}{r} +\ F2_{16} \\ \underline{\quad 3C_{16}} \\ +\ B58_{16} \\ 2D6_{16} \\ \hline 38B8_{16} \end{array}
\end{array}
$$

Детальніше процеси проведення арифметичних дій на прикладі двійкової системи числення представлені в віртуальному середовищі (Відео: «Арифметичні операції в двійковій системі числення»).

3. Індивідуальні практичні завдання

Завдання полягає у переведенні чисел між різними системами числення та виконання арифметичних операцій над ними. Результат виконання завдання оформлюється у вигляді текстового документу та, як було зазначено, має бути завантажений в віртуальне навчальне середовище. Рекомендовано, з метою зменшення обсягу пам'яті, в якості текстового редактора використовувати звичайний «Блокнот». Варіанти виконання завдань представлені в таблиці.

Порядковий номер студента в журналі	$x_{10} \to x_2$ та $x_2 \to x_{10}$	$x_{10} \to x_8$ та $x_8 \to x_{10}$	$x_{10} \to x_{16}$ та $x_{16} \to x_{10}$	$x_{16} \to x_2$ та $x_2 \to x_{16}$	$x_8 \to x_2$ та $x_2 \to x_8$	Виконати арифметичні операції додавання, віднімання та множення
1.	12585	9874	3658	A1B	1237	00101 та 0011
2.	6589	45634	20301	A1C	4561	00101 та 0111
3.	5214	32586	9850	A1D	7770	00101 та 1111

4.	45686	6542	36942	A1E	3213	00111 та 0011
5.	6589898	86135	2043	A1F	654	01111 та 0011
6.	5735	951396	65105	B2A	77756	11111 та 0011
7.	10014	216955	46925	B2C	7412	01101 та 0011
8.	12591	2561	21302	B2E	752	00011 та 0101
9.	2156	2583	4562	B2D	7631	10010 та 0100
10.	987455	74105	89622	B2F	1472	00101 та 1000
11.	45637	3658	2101	C3A	257	10101 та 1011
12.	32589	20305	9874	C3B	3672	10101 та 1001
13.	65422	98506	45630	C3D	7514	01101 та 0001
14.	86135	3694	32580	C3E	7535	11101 та 0011
15.	9513	20434	65424	C3F	103	01101 та 0111
16.	21695	65105	8613	D4A	3015	01011 та 1111
17.	25616	46925	951355	D4B	2507	01011 та 0101
18.	2583	21304	21695	D4C	406	11101 та 0001
19.	74105	45625	25614	D4E	6047	11101 та 0011
20	36587	8962	25831	D4F	707	11110 та 0111
21	20306	21014	74109	E5A	7074	11101 т 1111
22	9850	12582	58942	E5B	7205	11011 та 0001
23	36942	6589	12592	E5C	657	10111 та 0011
24	20434	52145	98036	E5D	7725	10111 та 0111
25	6510	45686	96032	E5F	6305	11011 та 1011
26	46922	65892	54101	F6A	756	11011 та 0101
27	21301	57352	6590	F6B	4550	10111 та 1101
28	45620	1001	254012	F6C	652	11011 та 1100

29	89622	12592	5089	F6D	7731	10111 та 1111
30	21010	2156	5034	F6E	6641	11010 та 1100

Контрольні запитання

1. Визначити поняття «система числення, основа системи числення». Що є доповненням числа «0» у двійковій с/ч? Що є основою двійкової с/ч?
2. Яку операцію необхідно виконати для перетворення цілої частини числа з однієї с/ч у іншу: а) ділення; б) віднімання; в) множення; г) додавання?
3. Які основні типи даних визначені в мові Python.

Практичне заняття № 3
«Встановлення і налаштування середовища розробки мовою Python»

Мета: оволодіння базисними поняттями щодо встановлення і налаштування середовища розробки мовою Python.

Після практичного заняття студенти (курсанти) повинні:

вміти: встановлювати та налаштовувати середовище розробки мовою Python.

знати: процедуру встановлення на ПК середовища розробки мовою Python.

План заняття:

1. Завантаження установника:
2. Встановлення інтерпретатора на комп'ютері:

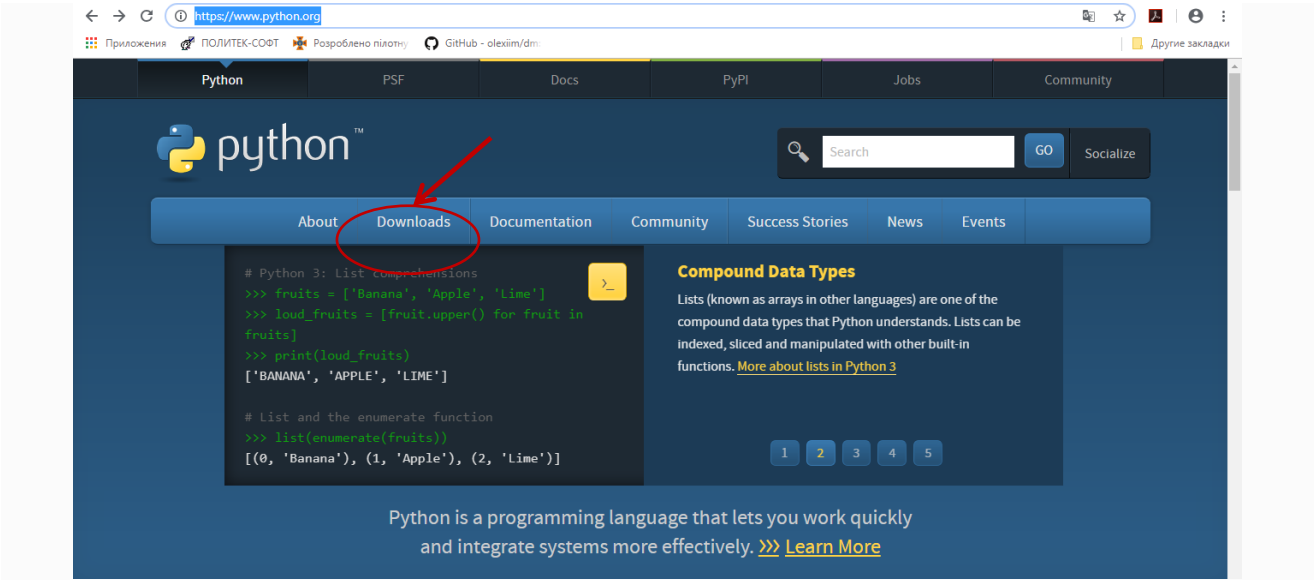
Завдання:

Якщо ви користуєтеся комп'ютером під керівництвом операційної системи Linux або MacOS, вам пощастило - інтерпретатор Python вже встановлений у вашій системі.

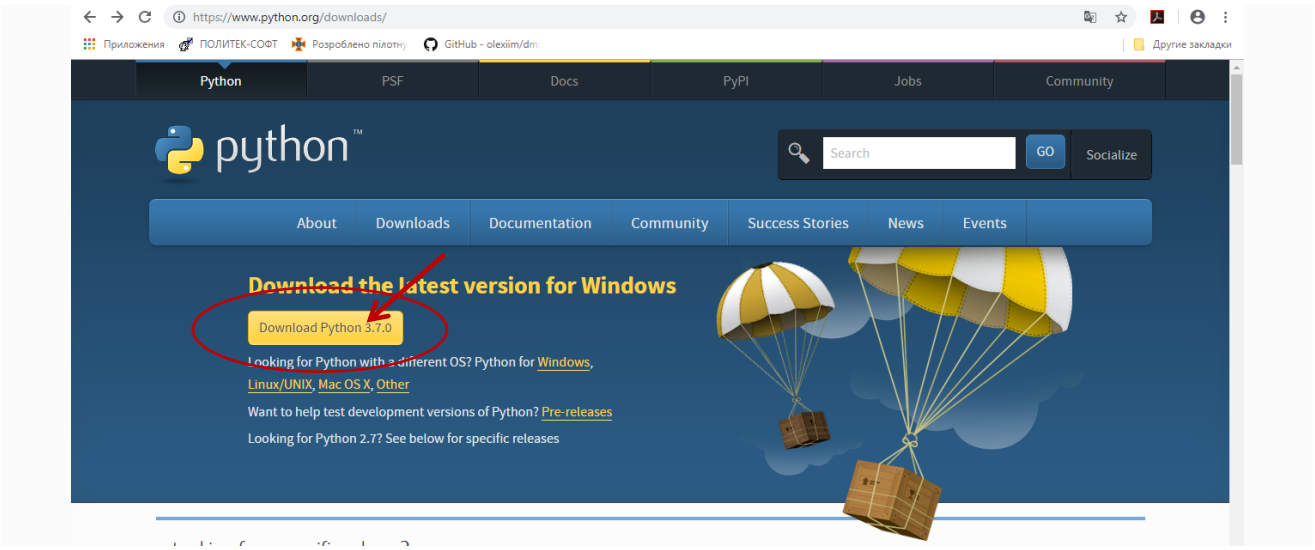
Якщо у Вас встановлено Windows, доведеться встановити інтерпретатор Python вручну.

I. Завантажуємо установник:

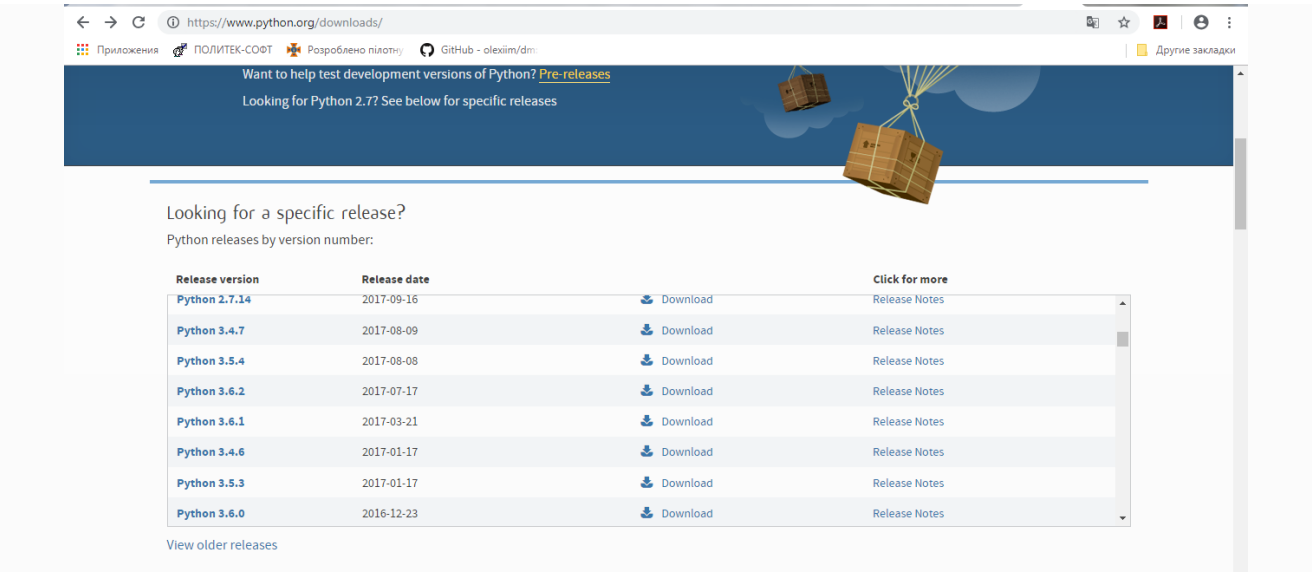
1. Заходимо на офіційний сайт Python (https://www.python.org/) та обираємо пункт меню "**Downloads**".



2. Натискаємо на кнопку для скачування версії 3.x.x (на момент написання тексту це 3.7.0).



(Якщо потрібна інша версія її можна знайти на цій же сторінці браузера прокрутивши бігунок лінійки вниз.)



3. Обираємо папку, куди Ви хочете зберегти установник.
4. Чекаємо на завантаження.

2. Встановлюємо інтерпретатор на комп'ютері:

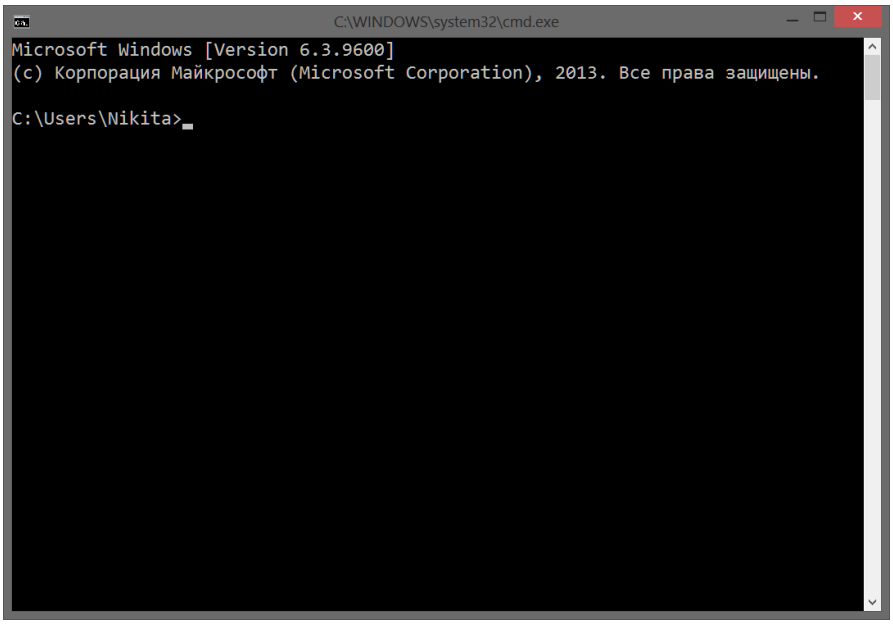
1. Після завантаження інсталяційного файлу необхідно його інсталювати на ПК. Процес інсталяції не передбачає жодної надскладної операцій, достатньо лише слідувати інструкціям інсталяційної програми. Запускаємо файл-установник із папки, де він був збережений.
2. Обираємо папку для установки. Раджу залишити папку "за умовчанням", так як наявність кирилиці або пробілів у шляху може в подальшому призвести до проблем із запуском деяких програм, які потребують Python для роботи.
3. На етапі "**Customize Python**" краще включити опцію "**Add python.exe to Path**" - це дозволить викликати інтерпретатор прямо за іменем python, не вказуючи повний шлях до програми. Зверніть увагу: ця можливість з'явиться лише після перезавантаження комп'ютера.
4. Чекаємо на завершення установки. **Finish**.

*Перезавантажуємо машину та **перевіряємо** працездатність інтерпретатора:*

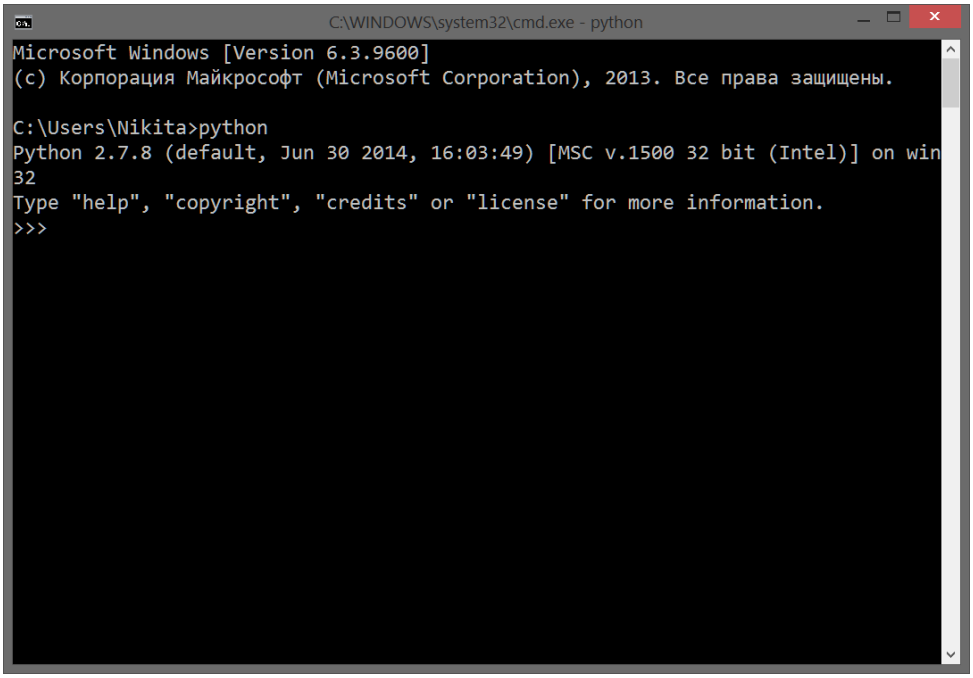
1. Запускаємо консоль. Ця дія знадобиться вам і далі, так як всі найпростіші програми працюють в консолі і ваші наступні лабораторні роботи також запускатимуться саме в ній:
2. для Windows: меню "пуск", виконати, вводимо "cmd" та тиснемо Enter.

3. для Linux та MacOS: знайдіть програму, яка називається "Термінал" або "Консоль" - точна назва може відрізнятися, в залежності від виду та версії вашої операційної системи.

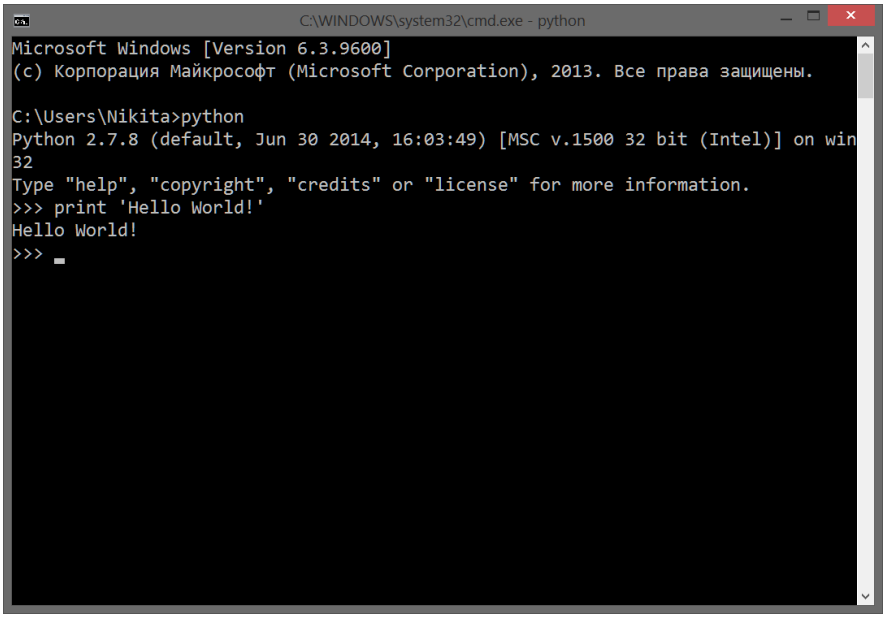
4. Перед вами з'явилося вікно з контрастним текстом. Це - командний рядок, в якому можна запускати програми та вводити різноманітні програми - фактично можна робити все, що дозволяє операційна система, але без графічного інтерфейсу.



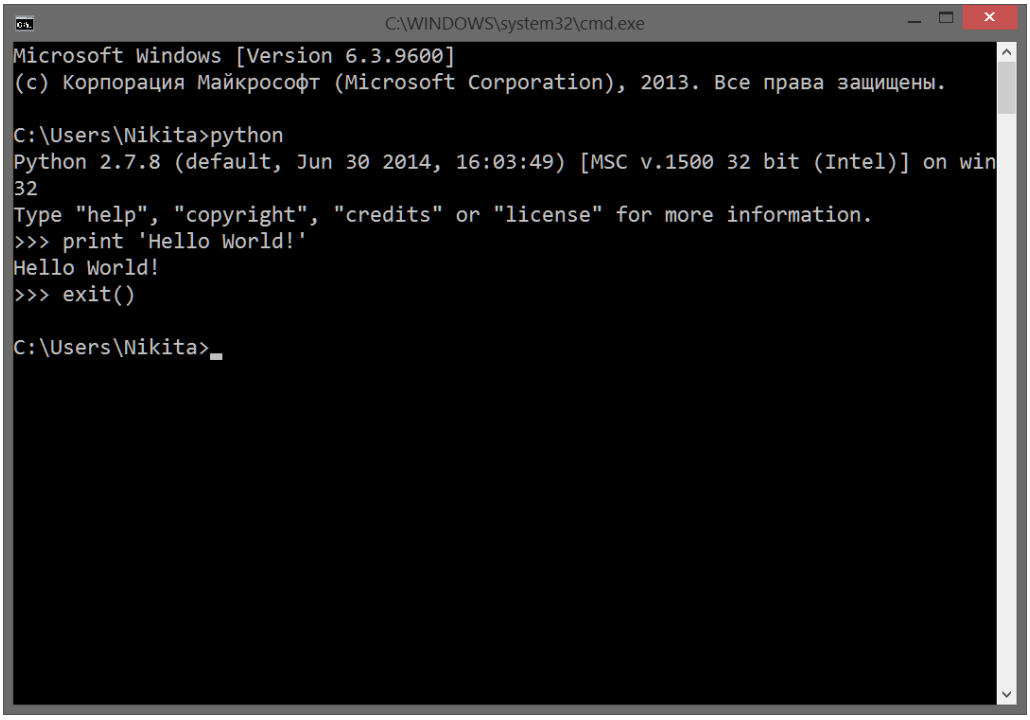
Якщо ви включили опцію "Add python.exe to Path", в консолі тепер можна просто набрати python та натиснути Enter – буде запущено сесію інтерпретатора Python в інтерактивному режимі. Це як ще одна консоль в консолі, лише яка сприймає команди мови Python замість команд операційної системи.



Вводимо просту команду: print 'Hello World!', тиснемо Enter. Інтерпретатор виконує команду: друкує нам у відповідь "Hello World!"



Для завершення інтерактивної сесії слід набрати і виконати команду exit().



Поздоровляю! Середовища розробки мовою Python коректно встановлене і налаштовано.

Практичне заняття № 4
«Розробка блок-схеми алгоритму розв'язання задачі»

Мета: ознайомитися з правилами побудови та визначення блок – схем алгоритмів для написання програм, сформувати вміння і навички роботи з графічними елементами та базовими структурами блок-схем. Об'єкт дослідження – блок–схема алгоритму, її графічні елементи та базові структури.

Після практичного заняття студенти (курсанти) повинні:

вміти: будувати алгоритм розв'язку поставленої задачі та візуалізувати його засобами Microsoft Visio.

знати: теоретичні основи опису алгоритмів за допомогою блок-схем.

Завдання

1. Вивчити теоретичні основи опису алгоритмів за допомогою блок-схем. Опрацювати приклади.
2. Відповідно до свого варіанту завдання виконати такі кроки
   - побудувати блок-схему алгоритму обчислення значень за даними варіантів завдань в Microsoft Visio,
   - сформувати блок-схему у середовищі Microsoft Visio, зберегти її в форматі Microsoft Visio та у текстовому документі.
3. Виконану роботу завантажити у навчальне середовище ЛДУ БЖД «Віртуальний університет».

Аудиторна робота

Розглянемо приклад алгоритму для знаходження середини відрізка за допомогою циркуля і лінійки.

Приклад 1. Алгоритм розподілу відрізка АВ навпіл:

- поставити ніжку циркуля в точку А;
- встановити розмах циркуля рівним довжині відрізка АВ;
- провести коло;
- поставити ніжку циркуля в точку В;
- провести коло;
- через точки перетину кіл провести пряму;
- відзначити точку перетину цієї прямої з відрізком АВ.

Приклад 2. Алгоритм додавання дробів

Даний алгоритм можна задати наступною послідовністю команд:

Вихідна інформація A, B, C, D; {скласти A/B і C/D}

Результат E,F; {E/F = A/B + C/D}

1. Обчислити Y = B*D; {Перейти до наступної команди}
2. Обчислити X1 = A*D; {Перейти до наступної команди}
3. Обчислити X2 = B*C; {Перейти до наступної команди}
4. Обчислити X = X1+X2; {Перейти до наступної команди}
5. Обчислити Z = НСД (X,Y); {Перейти до наступної команди}
6. Обчислити E = X div Z; {Перейти до наступної команди}
7. Обчислити F = Y div Z. {Закінчити роботу}.

Вихідна інформація алгоритму представлена чотирма цілими числами A, B, C, D. Це –чисельники і знаменники дробів, що додаються. Результат роботи алгоритму – числа E і F – чисельник і знаменник суми. Власне алгоритм складається з 7-ми команд, кожна з яких містить команду – виконати одну з арифметичних дій над цілими числами: додавання, множення, обчислення НСД і div (обчислення неповної частки).

Крім виконання арифметичної дії, кожна команда запам'ятовує результат як значення величини, зазначеної в лівій частині рівності, що входить у команду. Нарешті, кожна команда (у неявному виді) містить вказівку на наступну виконувану команду – перейти до виконання команди з наступним номером. Таким чином, алгоритм описує деталізований по кроках процес перетворення інформації.

Виконавець алгоритму не тільки виконує дії, але і запам'ятовує їхні результати. Для відображення цього факту в записі алгоритму ми використовуємо літерні позначення даних. Ці позначення називають іменами, а самі дані – величинами.

Приклад 3.Приклад опису алгоритму у словесній формі.

Задача: знайти модуль величини x (тобто значення |x|) і присвоїти це значення змінній y. Під час побудови алгоритму скористаємося визначенням модуля: | x| = x при x ≥ O і |x| = -x при x < 0.

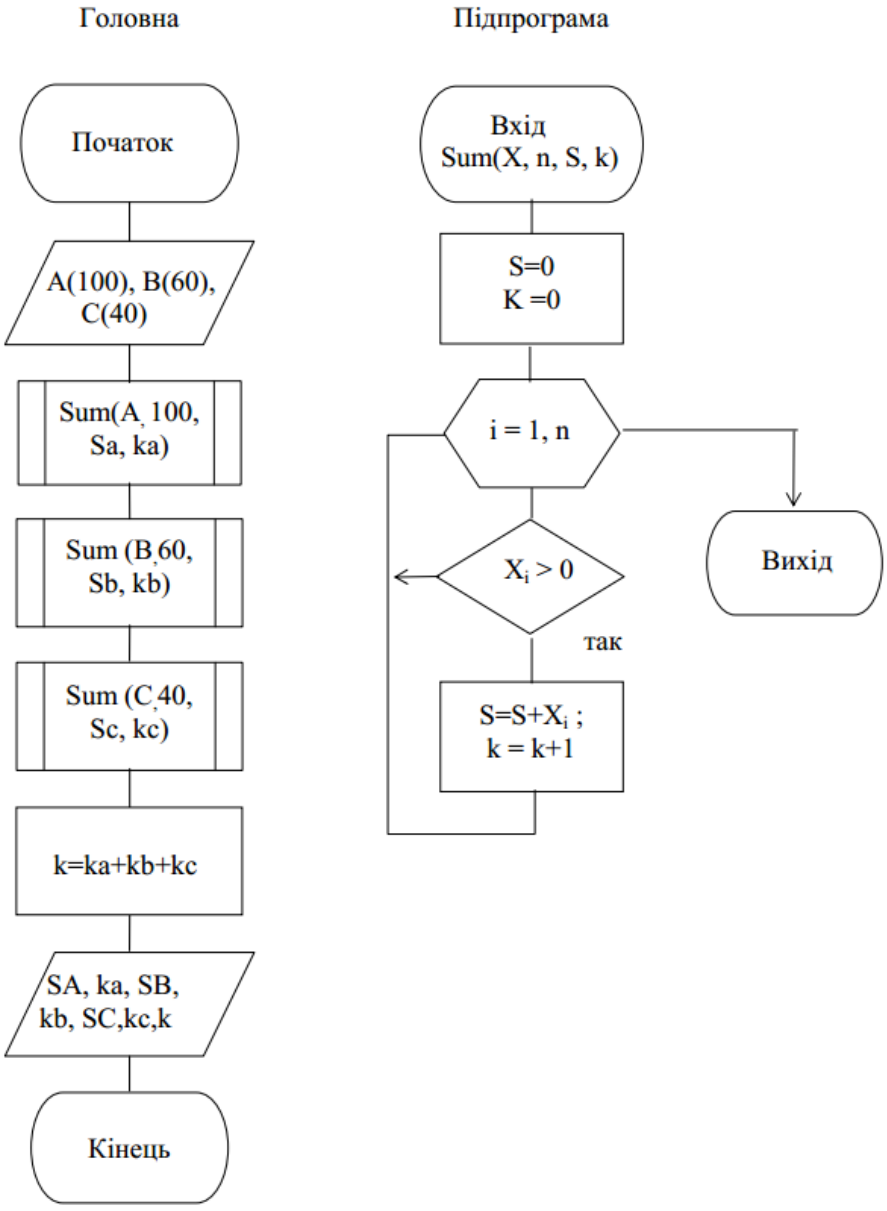
Алгоритм можна записати наступним чином

1 . Початок.
2. Ввести числове значення величини *x*.
3. Якщо $x \geq 0$, то *y* присвоїти значення *x,* інакше *y* присвоїти значення *-x*.
4. Вивести значення *y*.
5. Кінець.

Словесний запис найчастіше застосовується на початковому етапі вивчення алгоритмів і призначається для використання алгоритму людиною. Ця форма запису недостатньо наочна і її важко безпосередньо перекласти мовою програми.

Приклад 4.Приклад опису алгоритму у графічній формії.

Задача: відшукати суму та кількість елементів масивів A, B, C з використанням підпрограми



Індивідуальні практичні завдання

Таблиця 2.

Варіант	РОЗРАХУНКОВІ ФОРМУЛИ (Вихідні дані)	ВХІДНІ ДАНІ		
		x	y	z
1	2	3	4	5
1	$a=\begin{cases}\dfrac{1+\cos^2(x+y)}{\lvert 2y\rvert}, якщо - y>0\\ x^2, в\_інших\_випадках\end{cases}$	0.854	2.37	
2	$a=\begin{cases}\dfrac{\ln\lvert x\rvert}{\sqrt{\lvert x\rvert+\lvert y\rvert}}, якщо - x\le 0\\ x^3-y, в\_інших\_випадках\end{cases}$	-0.5	-1.42	
3	$b=\begin{cases}\dfrac{y^3}{x+y^3/(x+y)}, якщо - x>2\\ x^2, в\_інших\_випадках\end{cases}$	-3.45	4	
4	$b=\begin{cases}2^{-x}\sqrt{x+\sqrt[4]{\lvert y\rvert}}, якщо - y>0\\ x^3+6x^2+x+z, в\_інших\_випадках\end{cases}$	21.5	-3.21	2
5	$a=\begin{cases}\dfrac{1+\cos^2(x+y)}{\lvert 2x\rvert}, якщо - x>0\\ \sqrt[4]{\dfrac{e^{x-1}}{\sin z}}, в\_інших\_випадках\end{cases}$	3.91	1.6	0.5
6	$a=\begin{cases}\dfrac{1+z}{\lvert 2x\rvert}, якщо - x>0\\ \lvert x+z\rvert(\sin^2 z+tgz), в\_інших\_випадках\end{cases}$	-0.234		4.56
7	$k=\begin{cases}\sin\dfrac{x^2}{4}+\dfrac{e^{-z}}{\lvert x+y\rvert}, якщо - y>0\\ \sin(x^2-6z), в\_інших\_випадках\end{cases}$	1	2.3	-1
8	$a=\begin{cases}\dfrac{\lvert 2y\rvert}{1+\cos^2(x+y)}, якщо - x>0\\ a=\sqrt{y+\sqrt{x-1}}, в\_інших\_випадках\end{cases}$	124	-61	

9	$b=\begin{cases}\dfrac{\sin^2(x+1)-\cos^2(x+y)}{\lvert 2y\rvert}, якщо - y\neq 0\\ x(\sin(\ln z)+\cos^2 y), в\_інших\_випадках\end{cases}$	-2.367	0.56	101
10	$f=\begin{cases}a=e^{\lvert x-y\rvert}(tg^2 z+1), якщо - z\le 0\\ \sqrt{x^2-4yz}, в\_інших\_випадках\end{cases}$	2	1	0.67
11	$a=\begin{cases}\dfrac{\sqrt{y^2+z^2}}{\sin x}, якщо - x>0\\ \cos^2 arctg\dfrac{1}{x}, в\_інших\_випадках\end{cases}$	100	0,5	1,2
12	$a=\begin{cases}arctg\, x-\dfrac{x^2}{\sqrt[4]{\lvert x-y\rvert}}, якщо - x\neq y\\ \cos^2(x+y), в\_інших\_випадках\end{cases}$	45	-3.67	
13	$a=\begin{cases}\lvert y-x\rvert\dfrac{y-z/(y-x)}{1+(x-y)^2}, якщо - x\neq y\\ x^2-\sin x, в\_інших\_випадках\end{cases}$	137.51	1.567	-19
14	$a=\begin{cases}\dfrac{\cos^2(x+y)}{\lvert 2y\rvert}, якщо - y>0\\ y^x+\sqrt{\lvert x\rvert+\lvert y\rvert}, в\_інших\_випадках\end{cases}$	105	47	
15	$f=\begin{cases}\dfrac{\cos^2 y}{\lvert 2y\rvert}, якщо - y<0\\ \sqrt{x}+\dfrac{x^2}{x^2+y^2}, в\_інших\_випадках\end{cases}$	9	-167	
16	$a=\begin{cases}y+\dfrac{x}{y+\dfrac{x^2}{y+x^3/y}}, якщо - y\neq 0\\ x^2-arctg(x), в\_інших\_випадках\end{cases}$	-2	4.5	
17	$a=\begin{cases}\dfrac{1+tg(x+y)}{\lvert 2y\rvert}, якщо - y>0\\ \ln(\sqrt{e^{x-y}}+x^{\lvert y\rvert}+z), в\_інших\_випадках\end{cases}$	-0.23	0.998	100

18	$a=\begin{cases}1+\dfrac{z^2}{3+z^2/5}, якщо - x>0\\ \dfrac{x^2}{\sqrt{z^4}}, в\_інших\_випадках\end{cases}$	-2		199.75
19	$f=\begin{cases}\dfrac{\lvert\ln x+\cos y\rvert}{1+2\sin^2 x}, якщо - x>0\\ arctg(x^2-e^y), в\_інших\_випадках\end{cases}$	5.46	-0.978	
20	$s=\begin{cases}\ln(x^{\sqrt{\lvert y\rvert}})(x-\dfrac{y}{y+2}), якщо - x>0\\ \dfrac{x^2}{\sin(x)}, в\_інших\_випадках\end{cases}$	12.45	-3.1	
21	$a=\begin{cases}\dfrac{1+\cos^2(x+y)}{\lvert 2y\rvert}, якщо - y>0\\ \sqrt{10(\sqrt{x}+x^{y-1})}, в\_інших\_випадках\end{cases}$	100	-3.5	
22	$f=\begin{cases}\dfrac{1-\cos^2(x-y)}{\lvert 2y\rvert}, якщо - y>0\\ arctg^2 z+\lvert x+y\rvert, в\_інших\_випадках\end{cases}$	23.78	-78.23	45
23	$a=\begin{cases}\left\lvert\sin(x^3-y)\right\rvert, якщо - x>y\\ x^{\frac{y}{x}}-\sqrt[4]{y^{\lvert x\rvert}}, в\_інших\_випадках\end{cases}$	100	1000	
24	$f=\begin{cases}\dfrac{\cos^2(x+y)}{\lvert 2x\rvert}, якщо - x>0\\ e^{x+1}+arctg\ y, в\_інших\_випадках\end{cases}$	23.89	60	
25	$a=\begin{cases}\sqrt{\lvert y\rvert e^{y+\frac{x}{2}}}, якщо - x>0\\ \dfrac{x^2}{\sin y-\cos^2 x}, в\_інших\_випадках\end{cases}$	-5.78	12.1	
26	$a=\begin{cases}\dfrac{4y^2e^{3\sin x}}{3z^2}, якщо - x>0\\ x^2+6, в\_інших\_випадках\end{cases}$ $a=\dfrac{4y^2e^{3\sin x}}{3z^2}$	0.45	1	0.38

27	$a=\begin{cases}\dfrac{1}{\lvert 2y\rvert}, якщо - x>0\\ z\dfrac{\sqrt{y\ln x}}{1+tg^2x^2}, в\_інших\_випадках\end{cases}$	34	1.45	2
28	$a=\begin{cases}\dfrac{\ln(y+\sqrt{y+x})}{z+x^2+e^{x/2}}, якщо - y>0\\ x^4, в\_інших\_випадках\end{cases}$	-2.5	-0.1	14
29	$r=\begin{cases}\dfrac{x^2+4}{\sin^2 z+\dfrac{x}{2}}, якщо - z>0\\ \sin x^2, в\_інших\_випадках\end{cases}$	101.1		-0.945
30	$f=\begin{cases}\dfrac{x+\dfrac{y}{x^2}}{y(x-3)+e^{-x}}, якщо - y>0\\ x^2, в\_інших\_випадках\end{cases}$	12.175	45.54	

Контрольні запитання

1. Що таке алгоритм? Які існують способи опису алгоритмів?
2. Що таке блок-схема? Основні графічні елементи блок-схем, їх призначення.
3. Які існують правила оформлення блок-схем?
4. Який нормативний документ містить правила оформлення схем алгоритмів?
5. Які існують основні групи графічних форматів в Microsoft Visio? Які шаблони елементів застосовані для формування схеми алгоритму? Які засоби контролю за розмірами елементів передбачені Microsoft Visio? Які засоби вирівнювання і розподілу елементів застосовуються в Microsoft Visio? Як додати текст в діаграмі Microsoft Visio? Як здійснюється експорт фрагментів діаграм Microsoft Visio в текстовий редактор?

Практичне заняття 5
«Знайомство з Python-інтерпретатором. »

Мета: познайомитись з роботою Python-інтерпретатора.

Об'єкт дослідження: Python-інтерпретатор.

Після практичного заняття студенти (курсанти) повинні:

вміти: здійснювати елементарні математичні операції, використовуючи Інтерпретатор Python як калькулятор.

знати: пріоритети операцій Python-інтерпретатора , функції стандартного модуля ***math***

ПЛАН:

1. Інтерпретатор Python, як калькулятор.
2. Числа та операції над ними.

Аудиторна робота

Спробуємо повправлятися у простих математичних виразах в середині *Python*-інтерпретатора.

Спочатку йде рядочок, який починається символом решітки, це коментар до наступних після нього операцій. Він пояснює, що наступний код насправді робить. Потім йде рядочок коду одразу після >>> цим інтерпретатор запрошує програміста ввести його код. Після натискання ентеру (Enter), з'являється рядочок результату операції. Уже без *'>>>'*. Python

пробуємо додавати

```
>>> 1 + 1
2
 >>> 20 + 40
60
 >>> 12832 + 426872
439704
```

```
# пробуємо віднімати
                >>> 7 - 2
                5

 # множення
                >>> 3 * 2
                6
# тепер підносимо до квадрату
                >>> 3 ** 2
                9
```

 # print функція, вона пише передані їй аргументи на екран

```
# ви також можете друкувати змінні і результати математичних операцій в
реченні
# через кому ви можете передавати кілька аргументів print функції
                >>> print "year consists of ", 12*30 + 5, " days"
                year consists of 365 days

# ділення
                >>> 18 / 6
                3
 # зверніть увагу, залишок від ділення ігнорується, адже ми зараз працюємо
# з цілими числами
                >>> 20 / 6
                3
 # а тут уже передаємо десяткові дроби (float number), тому і результат
# відповідний, тобто тип даних результату в Python операції залежить від
# типу введених даних
                >>> 21.0 / 8.0
                2.625


# залишок від ділення
                >>> 20 % 6
                2
```

Базові операції в Pythoni. Порядок Операцій.

Пам’ятаєте таку річ як порядок (пріоритет) виконання математичних операцій у математичному виразі? В Python він існує також! І ось цей порядок, від більш пріоритетних операцій вгорі, до менш пріоритетних внизу:

1. дужки ()
2. піднесення до степеня **
3. множення *, ділення /, остача від ділення %
4. додавання+, віднімання –

Тут наведено кілька прикладів, що вказують на важливість порядку виконання операцій і те на скільки важливо пам’ятати про нього коли пишете власні математичні вирази на Python:

Операції в порядку зростання пріоритетів

Операція	Опис
Lambda	Лямбда-вираз
Or	Логічне АБО
And	Логічне І
Not	Логічне НЕ
in, not in	Перевірка належності
is, is not	Перевірка ідентичності
<, <=, >, >=, <>, !=, ==	Порівняння
|; ^; &	Побітове АБО; виключне АБО; І
<<, >>	Зсуви
+, –; *, /, %	Додавання, віднімання; множення, ділення
+x, –x	Унарні плюс і мінус
~x	Побітове НЕ
******	Піднесення до степеня
x[індекс]	Взяття елемента за індексом
x[від:до]	Виділення зрізу "від" і "до"
f(аргумент,...)	Виклик функції
(...)	Дужки або кортеж
[...]	Список або списокове включення
{ключ:дане,...}	Словник
`вираз,...`	Перетворення в рядок (` – наголос)

```
# комп'ютер спочатку порахує  4 * 3 і вже потім додасть 2
# це тому що операція множення має вищий пріоритет ніж додавання
>>> 2 + 4 * 3
14
```

У цьому випадку ми поставили дужки навколо операції додавання, тому те що в дужках виконається першим, а вже потім операція множення. Пам'ятаємо: дужки мають вищий пріоритет, ніж множення.

```
>>> (2 + 4) * 3
18
```

Ну і звичайно рівноцінні по пріоритету операції виконуються зліва на право.

```
>>> 3 - 30 - 2
-29
```

Дужки можуть змінювати порядок операцій, а також якщо маємо вкладені дужки ті що всередині - виконуються першими, ніж ті що ззовні.

```
>>> 3 - (30 - 2)
-25
```

Коментарі

Наступна річ, яку ви маєте знати, щоб рухатися далі до кілька-рядкових програм є ***Коментар***. У вашому Python інтерпретаторі введіть наступний рядок і подивіться на результат його виконання:

```
>>> # Hi, I am a Comment.
...
>>>
```

Так, нічого не з’явилося в рядочку результату. Лише порожня стрічка.

Ми щойно ввели рядочок-коментар.

Коментар – це частина коду, яка не запускається інтерпретатором. Можна сказати – ігнорується. ***В Python ви робите рядочок коментарем за допомогою символа решітки (#),*** який ставите попереду рядочка. Решітка коментує усе що йде після неї в рядочку, і нічого, що стоїть перед нею.

То ж ви можете ввести наступне:

```
# у цьому прикладі ми додаємо коментар вкінці рядочка з працюючим
кодом
```

```
>>> print "I love Ukraine" # this is my comment
I love Ukraine
```

Спробуйте ввести:

```
>>> # print "I live in Lviv"
...
>>>
```

у цьому випадку нічого не відбудеться оскільки символ решітки # стоїть із самого початку рядочка, тобто рядочок є повністю закоментований

а у цьому випадку ми забрали символ решітки перед нашим коментарем, і звичайно отримали повідомлення про помилку

```
>>> print "I love Ukraine" this is my comment
File "<stdin>", line 1
print "I love Ukraine" this is my comment
^
SyntaxError: invalid syntax
```

Чому на Вашу думку ми отримали повідомлення про помилку?

Індивідуальні практичні завдання

Написати код, що міститиме:

1. Три власних приклади, що допоможуть відобразити Ваше розуміння пріоритету операцій у мові ***Python.***
2. коментар стосовно пріоритету операцій у реалізованих Вами прикладах.

Контрольні запитання

1. Що таке коментар? Як з ним працює інтерпретатор.
2. Як позначається коментар у мові Python?
3. Який порядок виконання операцій у Python?
4. Як інтерпретатор запрошує програміста ввести його код?

Практичне заняття 6
«Реалізація алгоритмів послідовної (лінійної) структури на мові Python»

Мета: ознайомитися з алгоритмами послідовної (лінійної) структури, з процедурами запуску програм, які реалізують ці алгоритми на мові Python; знайомство з інтегрованим середовищем розробки - Integrated development environment (IDLE).

Об'єкт дослідження - алгоритми послідовної (лінійної) структури, процедури запуску програм, середовище програмування IDLE.

ПЛАН;

1. Python: запуск програми користувачем.
2. Числа та операції над ними.

Завдання

1. Вивчити теоретичні основи написання алгоритмів послідовної (лінійної) структури. Опрацювати приклади.

2. Відповідно до свого варіанту написати програму, яка, використовуючи складові модуля *math*, розраховує значення виразу (для тригонометричних функцій аргументи задано в радіанах, для введення даних з клавіатури та виведення даних на екран використовувати функції введення-виведення).

3. Засвоїти дві процедури запуску програми:

- в командному рядку;
- з інтерфейсу IDLE.

Програми мають бути записані в окремих файлах, які можуть виконуватися консольною командою python <ім'я файлу>.py.

4. Скласти звіт (створити *lab_rob_***Прізвище**.*docx* та помістити у нього усі зафіксовані, за допомогою клавіші ***PrtScr***, результати виконання кодів програм: *lab_rob1_1.py, lab_rob1_2.py, lab_rob1_3.py* та захистити його по роботі.

Аудиторна робота

Завдання 1. *Запуск програми в командному рядку DOS*

1. За допомогою текстового редактора «Блокнот» («Notepad») створити файл *lab_rob1_1.py,* який містить в собі наведений нижче код програми (інструкції мови). Зберегти даний файл на диску D:\ у папці КН. (Якщо папка КН відсутня на диску, створити її).

код програми (інструкції мови)

```
import sys
print(sys.platform)
print(2 ** 100)
x = 'Spam!'
print(x * 8)
print("Це робота курсанта/студента - ", "Вкажіть своє прізвище")
```

Пояснення:

Цей файл:

1. Імпортує модуль *sys* Python (бібліотеку додаткових інструментів), щоб пізніше отримати назву платформи ОС.
2. Чотири рази викликає функцію *print,* щоб відобразити результати.
3. Використовує змінну з ім'ям *x,* утворену в момент, коли їй присвоюють значення у вигляді об'єкта-рядка.
4. Виконує деякі операції над об'єктами.

2. Запустити файл *lab_rob1_1.py* на виконання за допомогою інтерпретатора Python в командному рядку DOS. Для цього необхідно виконати такі дії:

2.1. **Пуск (Start)) \ Все программы (Programs) \ Стандартные (Accessories) \ Командная строка (Command Prompt)**

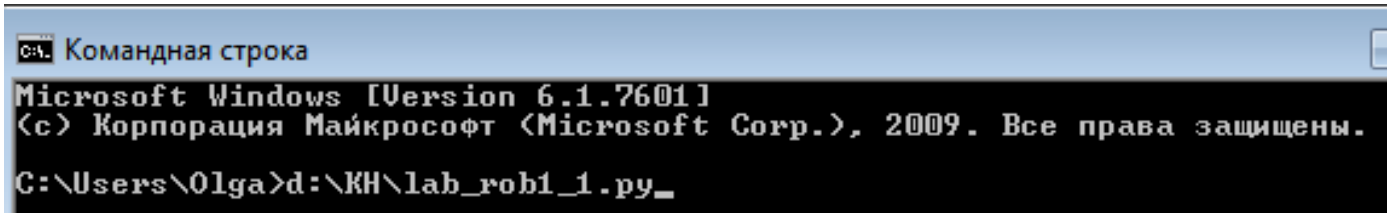
або

**Пуск (Start) \ Выполнить... (Run...) ** *cmd.*

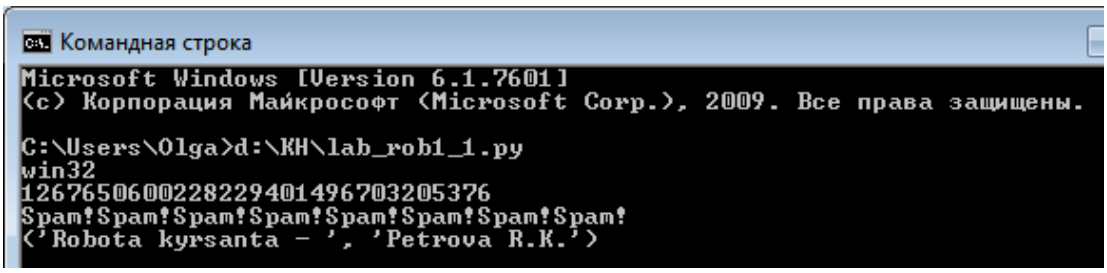
2.2. Вказати повний шлях до виконуваного файлу інтерпретатора:

D:\KH\lab_rob1_1.py

Підказка:



3. Зафіксувати результат виконання коду програми (клавіша PrtScr).



Завдання 2. *Запуск програми у середовищі IDLE.*

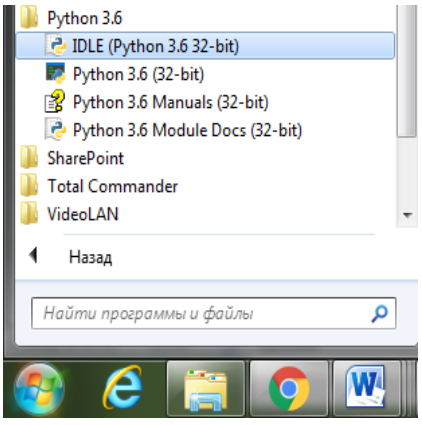
1. За допомогою текстового редактора IDLE створити текстовий файл *lab_rob1_2.py*, який містить в собі наведений нижче код програми (інструкції мови). Зберегти даний файл на диску D:\ у папці КН.

код програми (інструкції мови)

```
name = input("Як Вас звати? ")
surname = input("Вкажіть своє прізвище ")
day = input("Вкажіть який сьогодні день у родовому відмінку ")
print ()
print("Привіт,", name, surname, end="\n\n")
print("Гарної тобі", day, "!")
```

Підказка:

У меню Пуск, в розділі Python 3.x.x., є команда IDLE. Клацніть по ній мишею, щоб відкрити вікно середовища IDLE.



Щоб створити файл в головному вікні відкрийте меню **File (Файл)\ пункт New Window (Нове вікно)**, або натисніть Ctrl + N.

2. Запустити цей файл на виконання за допомогою інтерфейсу IDLE.

Підказка:

Для запуску сценарію, відкритого у вікні редагування, використовують меню «**Run**» цього вікна (пункт «**Run Module**») , або натисніть F5.

3. Зафіксувати результат виконання коду програми (клавіша ***PrtScr***).

Завдання 3. *Запуск програми у середовищі IDLE.*

1. За допомогою текстового редактора IDLE створити текстовий файл *lab_rob1_3.py,* який містить в собі код програми (інструкції мови). Зберегти даний файл на диску D:\ у папці КН.

Відповідно до свого варіанту напишіть код програми, яка, використовуючи складові модуля *math*, розраховує значення виразу $y = (\sin(\pi/2)/b)\cdot\sqrt{m}$. Передбачте, що m – ціле число (тип – int()), *b – дійсне число* (тип – float()).

Значення вхідних даних: m - відповідає Вашому № у списку

b - відповідає Вашому № у списку +15,5

Для введення даних скористайтесь функцією введення з клавіатури ***input***, для виведення даних на екран використайте функцію виведення даних ***print***).

Підказка:

- Спершу імпортуйте модуль math: import math
- Скористайтесь командою введення даних з клавіатури:

```
m=int(input("m="))
b=float(input("b="))
```

- Виклик функцій:

π	math.pi
sin ()	math.sin()
√	math.sqrt()

- виведення даних: print("y=", y)

2. Запустити цей файл на виконання за допомогою інтерфейсу IDLE.
3. Зафіксувати результат виконання коду програми (клавіша ***PrtScr***).

Індивідуальні практичні завдання:

Згідно варіанту (№ у списку групи) написати код програми, що може виконуватися консольною командою python <ім'я файлу>.py. (ім'я файлу - *lab_rob №ПР_Прізвище.py)*

Результат обчислень має виводитися на екран командою print.

Кожна програма має сприймати довільні значення, які задовольняють описаному в завданні формату, і передаються у вигляді аргументів командного рядка при виклику програми: python <ім'я файлу>.py

Формат введення даних та виведення результатів має чітко збігатися із вказаним.

Варіанти

№	РОЗРАХУНКОВІ ФОРМУЛИ (вихідні дані)	Значення вхідних даних
1	2	3
1	$Y=\sqrt{x^3+ax^2+bx+c}$, де a=x+0,52b; b=$e^{(cx2+1)}$	x=0,35; c=0,8
2	$Y=5\sin^2\lvert\ln(cx^3+1)\rvert$, де x=$\cos^2(a+b)$; b=$\sin^2(a)$	a=0,52; c=1,5
3	$Y=\ln(e^x+bx^2)$, де x=$\cos^2(a+b)$; b=$\sin^2(a)$	a=0,52
4	$Z=\frac{\sqrt{x^4+ax+b}}{\sqrt{x^4+ax+b}}$, де a=x^2+lg(0,08)+e^{-x}; b=x+4a	x=0,75
5	$C=\frac{D+\sqrt{x}}{\ln(\sqrt{R+E})}$, де D=12,5+ln(E); R=8D–x^4+1	x=1,08; E=0,7
6	$Y=(1+z)\frac{x+\frac{y}{x}}{a-\frac{1}{1+x}}$, де $y=e^{\sqrt{x^3+1,75z}}$; $z=5\cdot10^{-1,8}$	$x=0,8\cdot10^{-2}$
7	$Y=\frac{(a+b)^n}{1+\frac{x}{x^m+b^{m-n}}}$, де m=n–1; n=$e^{a/b}$; a=lg(0,083); b=e^a	x=0,81
8	$A=\frac{\alpha_1\beta_1-\alpha_2\beta_2}{\alpha_1-\alpha_2^2}$, де α_1=sin(0,18); $\beta_1=e^{\alpha_1}$; $\alpha_2=\ln(0,05)+e^{\alpha_1}$	β_2=1,7
9	$S=K_1a_1+K_2a_2-\beta$, де $a_1=a_0+\Delta a$; $a_2=T\sin(30^\circ)+\omega$; $\omega=e^{K_1+K_2}$	K_1=0,05; K_2=0,03; T=1; a_0=1; $\Delta a=0,1$
10	$Z=(a\sqrt{b}-c\sqrt{d})^2\frac{5,6}{a+b+c}$, де $a=\sqrt{(b-c)^3}$; b=a^2–1; d=$e^{(b^{\wedge}2-a)}$	c=1,0

1	2	3
11	$Y=\sin\frac{1}{x+0.2}+\lg(0,08e^x)$, де $x=-2,5a+b\sqrt{c}$; a=0,8c+bx	c=0,5; b=1
12	$Y=\frac{\cos^2 ax+b}{\sin^2 x}$, де $a=0,5c+e^x$; b=x^2–ac; c=0,8ln(0,072)	x=0,82
13	$Y=\frac{x+3a-K_1x}{K_2x+K_3x}$, де x=5a+K_1K_2	K_1=0,8; K_2=K_3=0,5; a=0,56
14	$Y=\frac{x^4-x^2-b}{x-b}+\frac{x^3-x-a}{x-a}$, де b=2^x–1; x=lg(0,005)+1	a=1,0
15	$Y=5\sin^2\lvert cx^3+1\rvert$, де x=cos^2(a+b); b=sin^2(a)	a=0,52; c=1
16	$Z=\frac{\sqrt{x^4+ax+b}}{\sqrt{x^4+ax+b}}$, де b=x+4a; a=-x^2+lg(0,08)+e^{-x}	x=0,75
17	$Y=(1+z)\frac{x+\frac{y}{x}}{a-\frac{1}{1+x}}$, де $y=e^{\sqrt{x^3+1,75z}}$; z=5·10$^{-1,8}$	x=0,8·10^{-2}; a=1,0
18	$A=\frac{\alpha_1\beta_1-\alpha_2\beta_2}{m\alpha_1-\alpha_2^2}$, де $\beta_1=e^{\alpha_1}$; $\alpha_2=\ln(0,05)+e^{\alpha_1}$; α_1=sin(π/2)	β_2=1,7; m=1
19	$Z=(a\sqrt{b}-c\sqrt{d})^2\frac{5,6}{a+b+c}$, де $a=\sqrt{(b-c)^3}$; b=a^2–1; $d=e^{(b^2-a^2)}$	c=1
20	$A=\frac{\alpha_1\beta_1-\alpha_2\beta_2}{\alpha_1-\alpha_2^2}$, де $\beta_1=e^{\alpha_1}$; $\alpha_2=\ln(0,05)+e^{\alpha_1}$; α_1=sin(π/3);	β_2=1,7
21	$y=\sqrt{x^3+ax^2+bx+c}$, де a=x+0,52b; b=$e^{(\alpha 2+1)}$	x=0,35; c=0,8
22	$y=\ln(e^x+bx^2)$, де x=cos^2(a+b); b=sin^2a	a=0,52
23	$C=\frac{D+\sqrt{x}}{\ln(\sqrt{R+E})}$, де R=8D – x^4+1; D=12,5+ln(E)	x=1,08; E=0,7

1. Усі зафіксовані, за допомогою клавіші ***PrtScr***, результати виконання кодів програм: *lab_rob1_1.py*, *lab_rob1_2.py*, *lab_rob1_3.py* помістити у файл *lab_rob*_**Прізвище.*docx*** та завантажити у віртуальне навчальне середовище.

2. Файл *lab_rob1_4.py* (індивідуальне завдання) завантажити у віртуальне навчальне середовище.

Контрольні запитання

1. Що таке алгоритм послідовної (лінійної) структури, програма послідовної (лінійної) структури?

2. Які основні типи даних визначені в мові Python.

Практичне заняття 7
«Реалізація алгоритмів розгалуженої структури (інструкція if) на мові Python»

Мета: ознайомитися з алгоритмами розгалуженої структури та їх реалізацією.

Об'єкт дослідження - умовний оператор (процедурна інструкція if), алгоритми розгалуженої структури.

ПЛАН

1. Динамічна типизація.
2. Умовна інструкція if.

Завдання

1. **Вивчити теоретичні основи написання алгоритмів розгалуженої структури. Опрацювати приклади:**

- набрати в консолі наведені нижче приклади;
- проаналізувати отримані результати;
- створити *lab_rob3_***Прізвище**.*docx* та помістити у нього усі зафіксовані, за допомогою клавіші ***PrtScr***, результати виконання кодів.

Аудиторна робота

Завдання 1.1. *Ознайомлення з роботою логічного оператора* **and**

```
print('and:')
print(False and False)
print(False and True)
print(True and False)
print(True and True)
print()
```

Завдання 1.2. *Ознайомлення з роботою логічного оператора* **or**

```
print('or:')
print(False or False)
print(False or True)
print(True or False)
print(True or True)
print()
```

Завдання 1.3. *Ознайомлення з роботою логічного оператора* **not**

```
print('not:')
print(not False)
print(not True)
print()
```

Завдання 1.4. *Логічні вирази*

```
a = True
b = False
c = True
f = a and not b or c or (a and (b or c))
print(f)
```

Завдання 1.5. *Приклади порівнянь*

```
a = 2; b = 5
print(a < b)        # менше
print(b > 3)        # більше
   print(a <= 2)    # менше або дорівнює
   print(b >= 7)    # більше або дорівнює
   print(a < 3 < b)# подвійне порівняння
   print(a == b)    # рівність
print(a != b)       # нерівність
print(a is b)       # ідентичність об‘єктів в пам‘яті
print(a is not b)   # a и b – різні об‘єкти
```

Індивідуальні практичні завдання

1. **Написати програму, яка розв‘язує наступне завдання:**

   (За допомогою текстового редактора IDLE створити файл lab_rob3_2_Прізвище.py, який міститиме в собі код програми (інструкції мови))

   Ввести дві змінних L_name та L_surname, що відповідають першим літерам Вашого імені та прізвища. Якщо вони співпадають, то надрукувати «True», інакше – «False». Організувати введення даних з клавіатури, виведення у консоль.

2. **Написати програму, яка розв‘язує наступне завдання:**

(За допомогою текстового редактора IDLE створити файл lab_rob3_3.py_Прізвище, який міститиме в собі код програми (інструкції мови))

Нехай задано числа a, b і c - довжина сторін трикутника. Якщо не можна побудувати трикутник із таким набором значень сторін, то надрукувати «0», інакше – «3», «2» або «1» в залежності від того, який це трикутник (рівносторонній, рівнобедрений або будь-який інший).

Організувати введення даних з клавіатури, виведення у консоль.

Підказка:

Слід пам’ятати, що не з кожних трьох відрізків можна побудувати трикутник. Як відомо, будь-яка сторона трикутника повинна бути менше суми двох інших.

Пам’ятка:

1. Усі зафіксовані, за допомогою клавіші ***PrtScr***, помістити у файл *lab_rob3_***Прізвище*.docx*** та завантажити у віртуальне навчальне середовище.
2. Файли *lab_rob3_2_Прізвище.py* та *lab_rob3_3_Прізвище.py* завантажити у віртуальне навчальне середовище.

Контрольні запитання

1. Що таке алгоритм розгалуженої структури?
2. В чому полягає різниця між умовними операторами з однією та двома гілками?
3. Що таке логічне висловлювання? Назвіть логічні операції, які використовують в логічних висловлюваннях при написанні програм на мові Python.
4. Який синтаксис має оператор розгалуження (множинного розгалуження)?

Практичне заняття 8
«Реалізація алгоритмів циклічної структури (інструкція while) на мові Python»

Мета: ознайомитися з алгоритмами розгалуженої структури та їх реалізацією.

Об'єкт дослідження - умовний оператор (процедурна інструкція **while**), алгоритми розгалуженої структури.

ПЛАН

1. Умовна інструкція while.
2. Обробка помилок.

Аудиторна робота

Завдання 1. **Вивчити теоретичні основи написання алгоритмів циклічної структури. Опрацювати приклади:**

1.1. За допомогою текстового редактора IDLE створити файл lab_rob_4_***Прізвище***.docx:

- набрати наведені нижче коди програм та проаналізувати отримані результати
- помістити у файл lab_rob_4_***Прізвище***.docx зафіксовані за допомогою клавіші ***PrtScr*** програмні коди та результати виконання кодів.

Приклад 1: ***(спробуйте спочатку ввести від'ємні числа)***

```
x =-1 0
while x <= 0: # повторювати, поки x не буде додатним
     x = int (input('Введіть додатне число: '))
   print ('Ви ввели число', x)
```

Приклад 2:

```
x = 0
while x < 10:
    x += 1
    if x == 5: # пропускаємо число 5
        continue
    print(' Поточне число дорівнює ', x)
```

Приклад 3:

```
a=2
while a!=10:
    a=a+1
    if a==6:
        continue
    print (a)
    if a==7:
        break
print ("Все!")
```

Приклад 4 (спробуйте спочатку ввести дані типу str (текстові символи)

```
n = input("Введіть ціле число: ")
try:
    n = int(n)
    print("Вдало")
except:
    print("Щось пішло не так")
```

1.2. У наступному циклі є помилка (*він виконуватиметься нескінченно*), **виправте цю помилку**. Програмний код та результат його виконання зафіксуйте за допомогою клавіші ***PrtScr та*** помістіть у файл lab_rob_4_***Прізвище***.docx

```
num = 1
while num < 10:
    print "Hello"
```

1.3. Проаналізуйте програмний код та розставте в потрібних місцях команди *break* та ***continue***. Програмний код та результат його виконання зафіксуйте за допомогою клавіші ***PrtScr та*** помістіть у файл lab_rob_4_***Прізвище***.docx

```
name = ''
while True:
    print('Who are you?')
    name = input()
    if name != 'Joe':
        continue
    print('Hello, Joe. What is the password? (It is a fish.)')
    password = input()
    if password == 'swordfish':
        break
print('Access granted.')
```

Завдання 2. **Напишіть програмну реалізацію наступного завдання:**

Послідовно вводяться ненульові числа. Визначити суму додатніх і суму від'ємних чисел. Закінчити введення чисел при введенні 0. Для вводу даних скористайтесь командою **input** (*для переводу з рядка в ціле число, використовувати функцію* **int ()).** Програмний код та результат його виконання зафіксуйте за допомогою клавіші ***PrtScr та*** помістіть у файл lab_rob_4_***Прізвище***.docx

Індивідуальні практичні завдання:

Скласти програму обчислення значення функції на відрізку [*x*0, *xk*] з кроком *h* за допомогою циклу *while*.

Згідно варіанту (№ у списку групи) написати код програми, що може виконуватися консольною командою python <ім'я файлу>.py. (ім'я файлу - lab_rob_4 ***Прізвище*.py**)

Результати обчислень мають виводитися на екран командою print. Кожне значення з нового рядка у форматі: ***y=a в точці x=b;*** де a=результат обчислення; b = значення при якому обчислювали функцію y.

Тобто наприклад: y=0,256 в точці x=1

y=0,325 в точці x=1,2

……..

Кожна програма має сприймати довільні значення, які задовольняють описаному в завданні формату, і передаються у вигляді аргументів командного рядка при виклику програми: python <ім'я файлу>.py.

№	$f(x)$	h; $[a; b]$
1	2	3
1	$y=\ln(x)$	h=0.1; a=1; b=1.5
2	$y=1+\ln^2(x)$	h=0.1; a=0.4; b=1.0
3	$y=1+e^x$	h=0.01; a=0.5; b=0.6
4	$y=e^{x^2}/2$	h=0.2; a=2; b=3
5	$y=\cos(x)\cdot e^{-x}$	h=0.2; a=1; b=2
6	$y=1/(1+e^{-x})$	h=0.2; a=3; b=4
7	$y=\sin(x)\cdot\sinh(x)$	h=1; a=1; b=5
8	$y=0.5+\sinh^2(x)$	h=0.2; a=2; b=3
9	$y=\sqrt{x}\cdot\cosh(x)$	h=0.2; a=3; b=4
10	$y=1/(1+\cosh^2(x))$	h=0.5; a=2; b=4
11	$y=\sqrt{x}\cdot\sinh(x)$	h=1; a=1; b=5
12	$y=e^{-x}\cdot\cosh(x)$	h=1; a=1; b=4
13	$y=\ln(x^2)$	h=0.1; a=1; b=1.4
14	$y=x+\ln(x)$	h=1; a=1; b=5
15	$y=1/(1+\sin(x))$	h=π/10; a==π/6; b==π/3
16	$y=\sin(x)+\sqrt{x}$	h=π/10; a==π/6; b==π/4
17	$y=x\cdot(1-\cos(x))$	h=0.1; a=0.4; b=0.8
18	$y=e^{x+3}\,sin(x)$	h=0.5; a=0; b=2
19	$y=\cos(x)\cdot\cosh(x)$	h=1; a=1; b=5
20	$y=e^{x+1}\cdot\sinh(x)$	h=1; a=1; b=4
21	$y=10^{-2}(5+4x)-e^{x^3+4}$	h=0.1; a=-3.4; b=-1.4
22	$y=4x^3+2^{5/4}xe^{-x}$	h=1.01; a=2.4; b=10.4
23	$y=9(x^3+3.2)\cdot\mathrm{tg}(x)$	h=0.2; a=1; b=2.4
24	$y=1.2e^{x^2}+x$	h=-0.05; a=-0.75; b=-1.5

Пам'ятка:

1. Усі зафіксовані, за допомогою клавіші ***PrtScr***, програмні коди та результати помістити у файл *lab_rob_4_***Прізвище*.docx*** та завантажити у віртуальне навчальне середовище.
2. Файли програмним кодом індивідуального завдання *lab_rob_4_Прізвище.py* завантажити у віртуальне навчальне середовище.

Контрольні запитання

1. Що таке цикл, для чого його використовують?
2. Як описується та виконується циклічна інструкція while?
3. Як можна організувати нескінченні цикли? Наведіть декілька прикладів і поясніть їх. Як можна вийти з нескінченних циклів? Що відбувається при запуску нескінченного циклу?
4. Чи може оператор циклу не мати тіла? Чому?
5. Для чого служать оператори переривання *break* та *continue*?

Практичне заняття 9
«Реалізація алгоритмів циклічної структури (інструкція for) на мові Python. »

Мета: ознайомитися з алгоритмами розгалуженої структури та їх реалізацією.

Об'єкт дослідження: умовний оператор (процедурна інструкція ***for***), алгоритми розгалуженої структури..

ПЛАН

1. Умовна інструкція for.
2. Обробка помилок.

Завдання

Завдання І. **Вивчити теоретичні основи написання алгоритмів циклічної структури. Опрацювати приклади:**

За допомогою текстового редактора IDLE створити файл lab_rob_5_***Прізвище***.docx:

- набрати наведені нижче коди програм та **проаналізувати** отримані результати
- помістити у файлlab_rob_5_***Прізвище***.docx зафіксовані за допомогою клавіші ***PrtScr*** програмні коди та результати виконання кодів.

Аудиторна робота

Приклад 1.1:

```
for x in range(11):
    if x == 5:
      continue
    print('Поточне число дорівнює ', x)
```

Приклад 1.2:

```
for x in range(3,11):
    if x == 5:
      continue
    print('Поточне число дорівнює ', x)
```

Приклад 1.3:

```
for x in range(2,11,3):
     if x == 5:
       continue
     print('Поточне число дорівнює ', x)
```

Приклад 2:

```
for i in range(10):
     for j in range(30):
         print('*', end='')
     print()
```

Приклад 3:

(Функція *reversed* дозволяє обходити послідовність в зворотному напрямку)

```
for i in reversed(range(5)):
     print(i)
```

Приклад 4:

```
word = "child" # рядок word
bag = ["knife","wallet", "pen","notebook"] # список bag

for letter in word:
    print (letter)

for item in bag:
    print (item )
```

*Приклад 5.1 (*метод ***isupper()***):

метод isupper()перевіряє великі букви символу (ЧИ З ВЕЛИКОЇ ВІН БУКВИ?), повертає True або False.

```
country="Ukraine"
for ch in country:
  if ch.isupper():
       print(ch)
```

Приклад 5.2 (метод ***islower ()***):

метод islower()перевіряє малі букви символу (ЧИ З малої ВІН БУКВИ?), повертає True або False.

```
country="Ukraine"
for ch in country:
  if ch.islower():
       print(ch)
```

Приклад 6.1:

```
for i in 'hello world':
  print(i * 2, end='')
```

Приклад 6.2 (оператор ***continue***):

```
for i in 'hello world':
  if i == 'o':
       continue
  print(i * 2, end='')
```

Приклад 6.3 (оператор ***break***):

```
for i in 'hello world':
  if i == 'o':
       break
  print(i * 2, end='')
```

Приклад 6.4 (оператор ***else***):

```
for i in 'hello world':
  if i == 'a':
       break
else:
  print('Букви a в стрічці немає')
```

Приклад 6.5 (оператор ***else***):

```
for i in 'hallo world':
  if i == 'a':
     break
else:
  print('Букви а в стрічці немає')
```

Індивідуальні практичні завдання

Завдання 1. **Напишіть програмну реалізацію наступного завдання:**

Знайдіть суму чисел на проміжку від N до M**.** Де N- відповідає Вашому № у списку+2, а M = N+25.

Для введення Вашого № у списку скористайтесь функцією введення з клавіатури ***input ()***, для виведення даних на екран використайте функцію виведення даних ***print ()***)

Програмний код та результат його виконання зафіксуйте за допомогою клавіші ***PrtScr та*** помістіть у файл lab_rob_5_***Прізвище***.docx

Завдання 2. **Напишіть програмну реалізацію наступного завдання:**

На вході: маємо рядок **dani,** що містить Ваше: прізвище, імя та по-батькові. На виході: абревіатуру, що складається з перших літер Вашого прізвища, імені та по-батькові**.**

Наприклад: **dani="Іванов Петро Степанович"**

Результат: **І.П.С.**

Програмний код та результат його виконання зафіксуйте за допомогою клавіші ***PrtScr та*** помістіть у файл lab_rob_5_***Прізвище***.docx

Контрольні запитання

1. Що таке цикл з параметром, формат його використання?
2. Що таке вкладені цикли?
3. Як працює інструкція *for*? Для організації яких циклів її застосовують ?

Практичне заняття 10
«Списки: одновимірні масиви» .

Мета: ознайомитися з особливостями визначення та використання одновимірних та двовимірних масивів, структурною організацією масивів та способів доступу до їх елементів.

Об'єкт дослідження: списки, масиви даних.

ПЛАН;

1. Списки
2. Масиви

Завдання

1. Ознайомитися з теоретичним матеріалом. Опрацювати приклади.
2. Відповідно до свого варіанту
   - визначити умови;
   - розробити програмний додаток, який розв'язує завдання
   - вивести на екран початковий та отриманий (отримані) вектори/матриці (або отриманий результат).
3. Скласти звіт і захистити його по роботі.

Для **завдань 1-3** *за допомогою текстового редактора IDLE створити файл lab_rob_3-1* Прізвище.docx:

- *набрати наведені нижче коди програм та* **проаналізувати** *отримані результати*
- *помістити у файл* 1аЬ_гоЬ_3-1_Прізвище.docx *зафіксовані за допомогою клавіші* PrtScr *програмні коди та результати виконання кодів.*

Аудиторна робота

Завдання І.

Приклад 1.1 *(*Для створення списку необхідно записати його елементи через кому у квадратних дужках)**:**

```
int_list = [1,2,3,5]
char_list = ['a',    'c', 'z', 'x']
empty_list = []

print('Список чисел:', int_list)
print('Список символів:', char_list)
```

```
print('Пустий список:', empty_list)
```

Приклад 1.2: (Значення елемента можна отримувати за його індексом Індексація починається з нуля.)

```
my_list = [5, 7, 9, 1, 1, 2] # Створення списку чисел
print (my_list)
print (my_list [0]);          # Виведення першого елемента
index = int (input ('Введіть номер елемента:'))
element=my_list[index]        # Отримання відповідного елемента
print (element)               # Виведення його значення на екран
```

Приклад 1.3: (Якщо використати від'ємні індекси, то обхід елементів розпочинається з останнього. Індекс останнього елемента списку - «-1», передостаннього - «-2».)

```
my_list = [5, 7, 9, 1, 1, 2] # Створення списку чисел
my_list_ch = ['5','7','9', 1','1','2'] # Створення списку символів
pre_last = my_list [-2] # Отримання передостаннього значення
print (pre_last)
result = my_list[0] + my_list[-1] # Обчислення суми першого і останнього значень
result_ch = my_list_ch[0] + my_list_ch[-1] # Обчислення суми першого і останнього значень
print(result)
print(result_ch)
```

Завдання 2. **Методи обробки списків**

Приклад 2.1: (Метод ***append*** додає значення в список)

```
my_list = []          # Створення порожнього списку
my_list.append(3)
my_list.append(5)
my_list.append(my_list[0]+my_list[1])
print(my_list)
```

Приклад 2.2: (Видалення елемента списку метод ***pop***)

```
colors=['red', 'orange', 'yellow', 'green', 'blue', 'purple']
print (colors.pop(1)) # Видалення другого елемента списку
print(colors)
print()
print(colors.pop())      # Видалення останнього елемента списку
print(colors)
print()
print (colors .pop(15)) # Видалення не існуючого елемента списку
```

Приклад 2.3: (Видалення елемента списку метод ***remove***)

```
my_list = [66.25, 333, 333, 1, 1234.5]
my_list.remove(333)
print(my_list)
my_list.remove(22225) # Видалення не існуючого елемента списку
print(my_list)
```

*Примітка: На від**міну** від методу **pop** метод **remove** при видаленні неіснуючого елемента видасть помилку*

```
Traceback (most recent call last):
  File "D:/OLGA/NAVCH/Python/Практичні/Приклади аудиторні/3-1.py", line 4, in <module>
    my_list.remove(22225)
ValueError: list.remove(x): x not in list
>>>
```

Приклад 2.4: **(Видалення** елемента списку **Оператор del)**

```
my_list = [5,1,5,7,8,1,0,-23] # Створення списку чисел
print(my_list)
del my_list[2] # Виведення списку. Оператор del видаляє заданий елемент
print(my_list)
```

Приклад 2.5: (Визначення **кількості** заданих елементів списку метод **count**)

```
my_list = [66.25, 333, 333, 1, 1234.5]
print(my_list.count(333), my_list.count(66.25),
my_list.count('x'))
print(my_list)
```

Приклад 2.6: (**Заміна** елемента списку метод **insert)**

```
my_list = [66.25, 333, 333, 1, 1234.5]
my_list.insert(2, "PPP")
print(my_list)
```

Приклад 2.7: (**Заміна** списку за допомогою зрізу)

```
my_list = [1, 2, 3, 4, 5, 6]
my list = my list[:2] + my list[3:]
print(my_list)
```

Приклад 2.8: (**Заміна** елемента списку)

```
my_list = [5, 1, 5, 7, 8, 1, 0, -23]    # Створення списку чисел
print(my_list)                  # Виведення списку
length = len(my_list)          # Отримання довжини списка
index = length
while not -length <= index < length:
    index=int(input("Введіть індекс ел-та списку від(від %d до %d):"
    % (-length, length - 1)))      # Введення нового значення
    value = int(input 'ведіть нове значення заданого ел-та: '))
    my_list[index]=value    # зміна елемента списку
print(my_list)
```

Завдання 3. **Виведення квадратів чисел зі списку**

```
my_list = [5, 1, 5, 7, 8, 1, 0, -23] # Створення списку чисел
for x in my_list:
     print('{}^2={}'.format(x, x ** 2))
```

Індивідуальні практичні завдання

Завдання 1. **Напишіть програмну реалізацію наступного завдання*:***

Маєте список L = [3, 6, 7, 4, -5, 4, 3, -1]. Визначте суму елементів списку L. ЯКЩО сума перевищує значення 2, то вивести на екран число елементів списку, інакше вивести значення суми.

Програмний код помістіть у файл іаЬ_гоЬ_3_1-4_Прізвище.ру та завантажте у віртуальне навчальне середовище.

Завдання 2. **Напишіть програмну реалізацію наступного завдання:**

Напишіть програму, яка запитує з введення з клавіатури вісім чисел та додає їх в список. На екран виводить їх суму, максимальне і мінімальне з них. Відсортуйте даний список та виведіть його на екран.

Для знаходження максимуму і мінімуму скористайтеся вбудованими в Python функціями max () і min ().

Програмний код помістіть у файл 1аЬ_гоЬ_3_1-5_Прізвище.ру та завантажте у віртуальне навчальне середовище.

Контрольні запитання

1. Що таке одновимірний масив? Для чого використовують одномірні масиви? Як їх описують в Python?
2. Як в програмі використати значення конкретного елемента одновимірного масиву?

Практичне заняття 11
«Зрізи. Двовимірні масиви. Генерація випадкових чисел»

Мета: ознайомитися з особливостями визначення та використання зрізів та двовимірних масивів, структурною організацією масивів та способів доступу до їх елементів.

Об'єкт дослідження: списки, зрізи, масиви даних, процедура генерації випадкових чисел (модуль random).

ПЛАН

1. Зрізи
2. Масиви.
3. Модуль random.

Завдання

1. Ознайомитися з теоретичним матеріалом. Опрацювати приклади.
2. Відповідно до свого варіанту
   - визначити умови;
   - за допомогою формул описати варіанти виконання необхідний дій;
   - розробити програмний додаток, який розв'язує завдання
   - організувати генерацію випадкових чисел (якщо треба, введення даних з клавіатури і виведення у консоль);
   - вивести на екран отриманий результат.
3. Скласти звіт і захистити його по роботі.

Захист роботи включає в себе демонстрацію працездатності програми на різних вхідних даних, демонстрацію трасування виконання програми.

Для **завдань 1-3** за допомогою текстового редактора IDLE створити файл lab_rob_3-3_***Прізвище***.docx:

- набрати наведені нижче коди програм та проаналізувати отримані результати
- помістити у файл lab_rob_3-3_***Прізвище***.docx зафіксовані за допомогою клавіші ***PrtScr*** програмні коди та результати виконання кодів.

Аудиторна робота

Завдання І. **Зріз списку**

Приклад 1.1 Зріз списку – отримання групи елементів за їхніми індексами:

```
my_list = [5, 7, 9, 1, 1, 2] # Створення списку чисел
sub_list = my_list[0:3]      # Отримання зрізу списку від 1 до 4.
print(sub_list)              # Виведення отриманого списку
print(my_list[2:-2])         # Виведення елементів списку від 3-го до передостаннього
print(my_list[4:5])          # Виведення ел-тів списку від 5-го до 6-го.
```

Результат :

[5, 7, 9]
[9, 1]
[1]

Приклад 1.2 Зріз з використанням кроку:

```
my_list = [5, 7, 9, 1, 1, 2]
sub_list = my_list[0:-1:2]   # Вибір кожного 2-го ел-та списку (починаючи з 1-ого), не включаючи останній елемент
print(sub_list)
print(my_list[2:-2:2])       # Виведення ел-тів списку від 3-го до передостаннього з кроком 2
print(my_list[-1:0:-1])      # Виведення ел-тів списка, крім першого, в зворотному порядку
```

Результат:

[5, 9, 1]
[9]
[2, 1, 1, 9, 7]

Приклад 1.3 Зріз із пропущеними параметрами:

Будь-який параметр зрізу можна пропустити (при умові дотримання правильного розміщення "**:**"). За замовчуванням початок списка – 0, кінець – довжина списка, крок – 1

Проаналізуйте отримані результати та допишіть коментарі до кожного рядка наведеного нижче програмного коду, тобто вкажіть в коментарях, які саме елементи списку буде виведено.

```
my_list=[1,2,3,4,5,6,7,8,9,10]
print(my_list[2:])        # Виведення елементів списку від 3-го до кінця
print(my_list[:-2])       # Виведення всіх ел-тів списку від початку до
                            передостаннього ел-та
print(my_list[::-1])      #.....
print(my_list[-1::])      #.....
print(my_list[:0:-1])     #.....
print(my_list[-1::-1])    #.....
print(my_list[-1:0:-1])   #.....
print(my_list[-2:0])      #.....
print(my_list[-4:-2])     #.....
print(my_list[-2:-7:-1])  #.....
print(my_list[0:-7:-1])   #.....
print(my_list[:-7:-1])    #.....
```

Завдання ІІ. Генератор випадкових списків (Модуль random)

Приклад 2.1 Генератор випадкових дійсних чисел (***random.uniform***):

Нехай задано два списки дійсних випадкових чисел від 0 до 5: [a_1, …, a_n] і [b_1, …, b_n], n=10. Написати програму їх формування. Вивести списки на екран.

```
import random
a=[random.uniform(0,5) for i in range(0,10)]
b=[random.uniform(0,5) for j in range(0,10)]
print(a);
print(b)
```

Результат:

```
======== RESTART: D:/OL/NAVCH/Python/Практичні/lab_rob_3_3-random.py ========
[4.456458938517761, 0.3774435355482836, 4.121398470246338, 2.4747469483070974, 3
.520103440774477, 4.162866280219348, 1.2012149258927274, 1.1327864865856263, 0.2
0253337697079432, 2.5839173970417533]
[2.0103389845599926, 4.329605090609384, 1.825402756317695, 3.6583325691005935, 4
.995628475751678, 0.3013995094347832, 1.3098782506022393, 4.003026161205636, 0.6
178276320200932, 3.895047689334226]
>>>
```

Приклад 2.2 Генератор випадкових цілих чисел ***(random.randint)***:

Нехай необхідно створити два списки [a_1, …, a_n] і [b_1, …, b_n], n=10, заповнених квадратами цілих випадкових чисел від 5 до 15. Написати програму їх формування. Вивести списки на екран.

```
import random
a=[random.randint(5,15)**2 for i in range(0,10)]
b=[random.randint(5,15)**2 for j in range(0,10)]
print(a);
print(b)
```

Результат:

```
[225, 121, 196, 81, 64, 81, 25, 144, 100, 196]
[49, 225, 144, 121, 100, 100, 49, 25, 64, 64]
>>>
```

Приклад 2.3 Генератор випадкових цілих чисел***(random.sample)***:

random.sample*(population, k) - список довжиною k з послідовності population.*

Нехай згенеровано список із цілих випадкових чисел та нулів [a1, …, an]. Написати програму визначення елементів, розміщених після першого нульового. Вивести на екран початковий та отриманий списки.

```
import random
arr=random.sample(range(-6,6), 12)
print("our random list: " , arr)
flag=0
while flag==0:
    if 0 in arr: # check if we have at list one zero in list
       first_zero_index=arr.index(0) #find index of first zero in list
       flag=1;
else:
     print("we have not at list one zero in list: ")
arr1=[]
if flag==1:
    for i in arr[first_zero_index:]:
       arr1.append(i)
print (arr1)
```

Результат:

```
our random list:  [-6, -1, -2, -5, -4, 4, 1, 2, 0, 3, -3, 5]
[0, 3, -3, 5]
>>>
```

Завдання ІІІ. Робота з матрицею. Двовимірний масив. Модуль random.

Приклад 3.1 Двовимірний масив**.** Генератор випадкових дійсних чисел (***random.randinit***):

Нехай задано список-матриця цілих випадкових чисел (додатних та від'ємних) [[*a11*, ..., *a1n*], ..., [*am1*, ..., *amn*]]. Написати програму, яка визначить список мінімальних елементів кожного рядка (результат записати в інший список). Вивести на екран початкову матрицю та отриманий список.

```
import random
n=3 # кількість стовпчиків
m=4 # кількість рядків
x=[0]*m #будуєм список з m нулів

for i in range(m): #будуєм порожню матрицю
    x[i]=[0]*n
y=[0]*m #будуєм список з m нулів
i=0
while i<m :
    j=0;
    while j < n :
        x[i][j] = random.randint(-5,5)
        j+=1
    y[i] = min(x[i][0:m])
    i+= 1
print('x= ', x); print('min = ', y)
```

Результат:

```
========= RESTART: D:\OL\NAVCH\Python\Практичні\lab_rob_3_
x=  [[4, 5, -1], [5, -5, -2], [0, -1, -3], [2, -3, -1]]
min =  [-1, -5, -3, -3]
>>>
```

Індивідуальні практичні завдання

Завдання 1 **Напишіть програмну реалізацію наступного завдання:**

Створіть список ***my_list*** із 10 цілих випадкових чисел на проміжку [0;30] та виведіть на друк, використовуючи зрізи:

а) кожен 3-ій елемент списку;

б) кожен 2-й елемент списку у зворотньому порядку;

в) елементи списку від 8-го до 5-го.

Програмний код помістіть у файл lab_rob_3_3-1_*Прізвище*.py та завантажте у віртуальне навчальне середовище.

Завдання 2 **Напишіть програмну реалізацію наступного завдання:**

Створіть список ***my_list*** із 10 цілих випадкових чисел на проміжку [-10;10].

Треба написати програму формування іншого списку, в якому всі від‘ємні елементи списку перенести в його початок, а всі інші – в кінець, зберігаючи початкове взаємне розміщення як серед від‘ємних елементів, так і серед інших элементів.

Програмний код помістіть у файл lab_rob_3_3-2_*Прізвище*.ру та завантажте у віртуальне навчальне середовище

Контрольні запитання

1. Для чого в програмах використовуються двовимірні масиви? Як їх описують в Python?

2. Скільки індексів характеризують конкретний елемент двовимірного масиву?

3. Як в програмі використати значення конкретного елемента двовимірного масиву?

4. Який індекс двовимірного масиву змінюється швидше при послідовному розміщенні елементів масиву в оперативній пам'яті?

5. Які функції виконує модуль random?

.

Практичне заняття 12
«Множина» .

Мета: ознайомитися з об‘єктом множина мови Python.

Об’єкт дослідження: множина, рядки символів.

ПЛАН

1. Рядки символів.
2. Множина.

Завдання

1. Ознайомитися з теоретичним матеріалом. Опрацювати приклади.
2. Відповідно до свого варіанту
   - визначити умови;
   - розробити програмний додаток, який розв‘язує завдання
   - організувати генерацію випадкових чисел (якщо треба, введення даних з клавіатури і виведення у консоль);
   - вивести на екран отриманий результат.
3. Скласти звіт і захистити його по роботі.

Захист роботи включає в себе демонстрацію працездатності програми на різних вхідних даних, демонстрацію трасування виконання програми.

Для **завдань 1-2** за допомогою текстового редактора IDLE створити файл lab_rob_3-4_***Прізвище***.docx:

- набрати наведені нижче коди програм та **проаналізувати** отримані результати
- помістити у файл lab_rob_3-3_***Прізвище***.docx зафіксовані за допомогою клавіші ***PrtScr*** програмні коди та результати виконання кодів.

Аудиторна робота

Завдання І. Перетин послідовностей у вигляді рядків

Приклад 1. Пошук перетину двох послідовностей у вигляді рядків:

І спосіб:

Розглянемо цикл ***for***, який вибирає елементи, загальні для двох рядків. Після того як цикл ***for*** виконано, змінна ***res*** буде посилатися на список, який містить всі однакові елементи, виявлені в ***seq1*** і ***seq2***:

```
seq1 = "spam"; seq2 = "scam"
res = [] # спочатку список пустий
for x in seq1: # виконати обхід першої послідовності
    if x in seq2: # загальний елемент?
        res.append(x) # Додати в кінець результата
print (res)
```

Результат:

```
=== RESTART: D:/
['s', 'a', 'm']
```

II спосіб:

Використаємо генератор списку:

```
seq1 = "spam"; seq2 = "scam"
res = [] # спочатку список пустий
[res.append(x) for x in seq1 if x in seq2]
print (res)
```

Результат:

```
=== RESTART: D:/
['s', 'a', 'm']
```

III спосіб:

Використаємо множини для визначення перетину послідовностей у вигляді рядків:

```
x=set("spam"); y = set("scam")
z=x & y # Перетин множин
print(z)
```

Результат:

```
=== RESTART: D:/OL/
{'a', 'm', 's'}
```

Завдання 2. **Множини.**

Приклад 2.1. Створення множин – екземплярів ***класу set:***

```
my_set={1, 5, 3, 9} # множина цілих чисел
print(my_set)

basket={'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
print(basket)
```

Результат:

```
== RESTART: D:/OL/NAVCH/Python/Практи
{1, 3, 5, 9}
{'orange', 'apple', 'pear', 'banana'}
```

Приклад 2.2. Створення множин на основі генераторів множин:

```
number_set={i + j for i in range(10) for j in range(5)}
print(number_set)

char_set={x for x in 'abracadabra' if x not in 'abc'}
print(char_set)
```

Результат :

```
== RESTART: D:/OL/NAVCH/Python/Практичні/Прикл
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
{'d', 'r'}
```

*Приклад 2.3. Використання **конструктора frozenset** (формування незмінної множини).*

Множини можуть включати об'єкти тільки незмінних типів. Якщо необхідно зберегти одну множину всередині іншої, створюють множину за допомогою вбудованої функції ***frozenset,*** яка діє так само, як функція ***set***, але формує незмінну множину.

```
empty_set=set()
print(empty_set) # Пуста множина

empty_frozenset=frozenset() # Пуста незмінювана множина
print(empty_frozenset)

my_frozenset=frozenset([4,1,3,8])
my_set=set(my_frozenset)
print(my_set)
print(my_frozenset)
```

Результат :

```
== RESTART: D:/OL/NAVCH/Python/Практичні/Прикла
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
set()
frozenset()
{8, 1, 3, 4}
frozenset({8, 1, 3, 4})
```

*Приклад 2.4. Операції з множинами (**set, frozenset**).*

Проаналізуйте отримані результати

```
my_set = {4, 5, 1, 2}
print('len({})={}'.format(my_set, len(my_set))) # К-сть елементів м-ни
print()
```

```
print(4 in my_set) # Перевірка входження елемента в м-ну
print(3 not in my_set)
print(9 in my_set)
print()

print({3, 4, 5}.isdisjoint({8, 1, 0})) # Чи перетинаються множини
print({3, 4, 5}.isdisjoint({1, 2, 3}))
print()

print({1,7,9}.issubset({1,2,3,7,9})) # Перевірка включення однієї мн-ни в іншу
print({1, 7, 9} <= {1, 2, 3, 7, 9})
print({1, 7, 9, 2, 3} <= {1, 2, 3, 7, 9})
print()

print({1, 7, 9} < {1, 2, 3, 7, 9}) # Перевірка чіткого включення
print({1, 7, 9, 2, 3} < {1, 2, 3, 7, 9})
print()

print({1,2,3,4}.issuperset({1, 2})) # Перевірка включення однієї мн-ни в іншу
print({1, 2, 4, 4} >= {1, 2})
print({1, 2, 3, 4} >= {1, 2, 3, 4})
print()

print({1, 2, 4, 4} > {1, 2}) # Перевірка чіткого включення
print({1, 2, 3, 4} > {1, 2, 3, 4})
print()

print({1, 3}.union({2, 3, 4})) #Об'єднання множин
print({1, 3} | {2, 3, 4})
print()

print({1, 3}.intersection({2, 3, 4})) # Перетин множин
print({1, 3} & {2, 3, 4})
print()


print({1, 2, 3, 4}.difference({3, 4, 5})) # Різниця множин
print({1, 2, 3, 4} - {3, 4, 5})
print()

print({1,2,3,4}.symmetric_difference({3,4,5,6})) # Симетрична різниця
print({1, 2, 3, 4} ^ {3, 4, 5, 6})
print()

my_set = set('chars') # Копіювання множини
copy = my_set.copy()
print(copy)
```

Приклад 2.5. Операції над множинами, з використанням методів, які мають в якості аргументів будь-які ітерабельні об'єкти.

Операції над множинами, записані у вигляді бінарних операцій, вимагають, щоб другий операнд операції теж був множиною, і повертають множину того типу, якою була перше множина

```
print(frozenset('abc').union(frozenset('cdef'))) # коректно
print(frozenset('abc') | frozenset('cdef')) # коректно
print(frozenset('abc').union('cdef')) # коректно
print(frozenset('abc') | 'cdef') # помилка
```

Проаналізуйте отримані результати

Результат :

```
frozenset({'d', 'b', 'c', 'a', 'e', 'f'})
frozenset({'d', 'b', 'c', 'a', 'e', 'f'})
frozenset({'d', 'c', 'b', 'a', 'e', 'f'})
Traceback (most recent call last):
  File "D:/OL/NAVCH/Python/Практичні/Приклади аудиторні/operacii_z_mnozhynamy_3-4.py",
line 56, in <module>
    print(frozenset('abc') | 'cdef') # помилка
TypeError: unsupported operand type(s) for |: 'frozenset' and 'str'
>>>
```

Індивідуальні практичні завдання

Завдання 1. **Напишіть програмну реалізацію наступного завдання:**

Задано множину символів від 'a' до 'z':
Aa Bb Cc Dd Ee Ff Gg Hh Ii Jj Kk Ll Mm Nn Oo Pp Qq Rr Ss Tt Uu Vv Ww Xx Yy Zz

Скласти програму, яка визначає і виводить на екран:

1. елементи цієї множини в алфавітному порядку.
2. елементи цієї множини в зворотньому порядку

Програмний код помістіть у файл *lab_rob_3_4-1_**Прізвище.**py* та завантажте у віртуальне навчальне середовище.

Завдання 2 **Напишіть програмну реалізацію наступного завдання:**

Створіть дві множини A та B:
- A - множина цілих чисел від «10» до «50»
- B - множина квадратів цілих чисел на проміжку від «1» до «10»

Скласти програму, яка визначає і виводить на екран:

1. Перетин цих множини.
2. Об`єднання цих множини.
3. Різницю множин B та A.
4. Чи належить число 8 множині B?

Програмний код помістіть у файл *lab_rob_3_4-2_**Прізвище.**py* та завантажте у віртуальне навчальне середовище.

Практичне заняття 13
«Множина. Методи роботи з об'єктами множини.»

Мета: ознайомитися з методи роботи з об'єктами множини мови Python.

Об'єкт дослідження: методи роботи з об'єктами множини.

Завдання

1. Ознайомитися з теоретичним матеріалом. Опрацювати приклади.
2. Відповідно до свого варіанту
   - визначити умови;
   - розробити програмний додаток, який розв'язує завдання
   - вивести на екран отриманий результат.
3. Скласти звіт і захистити його по роботі.

Захист роботи включає в себе демонстрацію працездатності програми на різних вхідних даних, демонстрацію трасування виконання програми.

Для **завдань 1-4** за допомогою текстового редактора IDLE створити файл lab_rob_3-5_***Прізвище***.docx:

- набрати наведені нижче коди програм та **проаналізувати** отримані результати
- помістити у файл lab_rob_3-3_***Прізвище***.docx зафіксовані за допомогою клавіші ***PrtScr*** програмні коди та результати виконання кодів.

Аудиторна робота

Приклад 1. Для обробки об'єктів-множин існують такі методи: ***add*** - вставляє новий елемент в множину, ***update*** - виконує об'єднання, ***remove*** - видаляє елемент за його значенням (викличте функцію ***dirset***, щоб отримати повний перелік всіх доступних методів):

```
my_set = {1, 3, 5}
my_set.update({2, 3, 4}) # my_set |= {2,3,4}
print(my_set)
my_set.intersection_update({0,1,2,3,10}) # my_set &= {0,1,2,3,10}
print(my_set)

my_set.difference_update({1}) # my_set -= {1}
print(my_set)

my_set.symmetric_difference_update({3, 4}) # my_set ^= {3, 4}
print(my_set)
```

```
my_set.add(5) # my_set |= {5}
print(my_set)

my_set.remove(2) #my_set-={2},виводить KeyError, якщо елемента немає
print(my_set)

my_set.discard(2) # my_set -= {2}
print(my_set)

print(my_set.pop())
print(my_set)

my_set.clear()
print(my_set)
```

Результат:

```
 RESTART: D:/OL/NAVCI
-5.py
{1, 2, 3, 4, 5}
{1, 2, 3}
{2, 3}
{2, 4}
{2, 4, 5}
{4, 5}
{4, 5}
4
{5}
set()
```

Приклад 2. Сформувати і вивести множину символів в алфавітному порядку

```
a = set('qwertyuiopasdfghjklzxcvbnm') # формуємо множину
print (a)      #виведення множини
b=list(a)      # перетворюємо множину у список (її не можна сортувати)
b.sort ()      #сортуємо список
print (b)      #виведення
```

Результат:

```
{'v', 'o', 'g', 'x', 'r', 'q', 'n', 'z', 'y', 'd', 'a', 'w', 'j', 'f', 'u', 'm',
 'k', 's', 'p', 'c', 'e', 'i', 'b', 't', 'h', 'l'}
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
>>>
```

Приклад 3. Застосування операцій над множинами, які використовують до списку людей – службовців гіпотетичної компанії:

```
>>> engineers = {"bob", "sue", "ann", "vic"}
>>> managers = {"tom", "sue"}
>>> "bob" in engineers # bob – інженер?
True
```

```
>>> engineers & managers # хто одночасно є інженером і менеджером?
{"sue"}

>>> engineers | managers # Всі співробітники з обох категорій
{"vic", "sue", "tom", "bob", "ann"}

>>> engineers - managers # Інженери, які не є менеджерами
{"vic", "bob", "ann"}

>>> managers - engineers # Менеджери, які не є інженерами
{"tom"}

>>> engineers > managers # чи всі менеджериы є інженерами?
False # (надмножина)

>>> {"bob", "sue"} < engineers # Обидва співробітника є інженерами?
True

>>> (managers | engineers) > managers # Множина всіх співробітників є
                                      надмножиною менеджерів?
True

>>> managers ^ engineers # Співробітники, які належать до однієї категорії
{"vic", "bob", "ann", "tom"}

>>> (managers | engineers) - (managers ^ engineers) # Перетин!
{"sue"}
```

Приклад 4. *Задано два слова. Для кожної літери першого слова (в тому числі для літер, які повторюються в цьому слові) визначити, чи входить літера до складу другого слова. Наприклад, якщо задано слова «інформація» і «процесор», то для літер першого із них відповідь повинна мати вигляд: ні ні ні так так ні ні так ні ні.*

Спосіб І

```
import itertools
sl1=input('Введіть слово 1 = ')
sl2=input('Введіть слово 2 = ')
m=len(sl1)
n=len(sl2);i=0; L=[]
while m!=i:
    fl=0;
    for j in itertools.count(start=0, step=1):
        if j>=n: break;
        if j<n:
            if sl1[i]==sl2[j]: fl=1
    if fl==1:
        L.append('да ')
    else:
        L.append('нет ')
    i+=1
a=list(L)
pp=''.join(a)
print(pp)
```

Результат:

```
 RESTART: D:/OL/NAVCH/Python/Практичні
-5-2.py
Введіть слово 1 = інформація
Введіть слово 2 = процесор
нет нет нет да да нет нет да нет нет
```

Спосіб ІІ

```
sl1=input('Введіть слово 1 = ')
sl2=input('Введіть слово 2 = ')
m=len(sl1)
i=0; L=[]
while m!=i:
    fl=0;
    if sl1[i] in sl2: fl=1
    if fl==1:
        L.append('да ')
    else:
        L.append('нет ')
    i+=1
a=list(L)
pp=''.join(a)
print(pp)
```

Результат:

```
 RESTART: D:/OL/NAVCH/Python/Практичні
-5-2.py
Введіть слово 1 = інформація
Введіть слово 2 = процесор
нет нет нет да да нет нет да нет нет
```

Контрольні запитання

1. Що таке множина?
2. Як зберігається множина в пам'яті ЕОМ? Який максимальний обсяг оперативної пам'яті можна відвести під зберігання однієї множини?
3. Які операції можна виконувати над множинами?
4. Як додати елемент в множину?
5. Як видалити елемент з множини?
6. Як вивести елементи множини? Як підрахувати кількість елементів у множині?

Практичне заняття 14
«Функції користувача. Рекурсивні функції» .

Мета: ознайомитися з принципами побудови функцій користувача на мові Python, з використанням локальних і глобальних змінних.

Об'єкт дослідження: функції користувача (основні складові функцій, оголошення та опис функцій), оператор форматування %..

ПЛАН;

1. Функції користувача.
2. Рекурсивні функції.

Завдання

1. Ознайомитися з теоретичним матеріалом. Опрацювати приклади.
2. Відповідно до свого варіанту
   - визначити умови;
   - розробити програмний додаток, який розв'язує завдання
   - вивести на екран отриманий результат.
3. Скласти звіт і захистити його по роботі.

Захист роботи включає в себе демонстрацію працездатності програми на різних вхідних даних, демонстрацію трасування виконання програми.

Для **завдань I** за допомогою текстового редактора IDLE створити файл lab_rob_3-6_***Прізвище***.docx:

- набрати наведені нижче коди програм та **проаналізувати** отримані результати
- помістити у файл lab_rob_3-6_***Прізвище***.docx зафіксовані за допомогою клавіші ***PrtScr*** програмні коди та результати виконання кодів.

Для **завдань II та III** за допомогою текстового редактора IDLE створити файл *lab_rob_3-6_1-Прізвище.py* та *lab_rob_3_6-2_**Прізвище.py*** та помістити їх у віртуальне середовище.

Аудиторна робота

Завдання I

Приклад 1.1 Оголошення функції **hello_world**

```
def hello_world():
    print('Hello, World!')
```

```
hello_world()   # Виклик функції
```

Приклад 1.2. Оголошення функції **print_numbers**

а) ***Формальний*** *параметр функції*

```
def print_numbers(limit):
     for i in range(limit):
          print(i)
print_numbers(10) # Виклик функції print_numbers, її формальний параметр
                  limit замінюють фактичним параметром 10
```

б) ***Фактичний*** *параметр функції*

```
def print_numbers(limit):
     for i in range(limit):
          print(i)
n = int(input('Введіть n: '))
print_numbers(n) # Виклик функції з фактичним параметром n print_numbers
```

c)

```
def print_numbers(limit):
     for i in range(limit): print(i)
def main():
     n=int(input('Введіть n: '))
     print_numbers(n)
main() # Виклик функції
```

Приклад 2. Оголошення функції з наперед невідомою кількістю аргументів:

Щоб оголосити функцію з будь-якою кількістю позиційних та іменованих аргументів, треба в списку формальних параметрів поставити два символи **

```
def unknown (* args):
    for argument in args:
        print argument
unknown ( "hello", "world") # надрукує обидва слова, кожне з нового рядка
unknown (1,2,3,4,5)         # надрукує всі числа, кожне з нового рядка
unknown ()                  # нічого не виведе
```

Приклад 3. Оголошення функції з аргументами - ключовими словами:

Аргументи - ключові слова використовуються при виконанні функції. Завдяки ключовим словам, ви можете задавати довільний (тобто не такий яким він описаний при створенні функції) порядок аргументів.

Приклад 3. 1

```
def person(name, age):
    print (name, "is", age, "years old")


person(age=23, name="John") # Хоча в описі функції першим аргументом йде ім'я, ми можемо викликати функцію так
```

Результат:

```
John is 23 years old
>>>
```

Приклад 3. 2 Функція, яка має три аргументи

```
def info(object, color, price):
     print('Обєкт:', object)
     print('Колір:', color); print('Ціна:', price)
     print()
info('ручка', 'синий', 1) # Виклик функції, передача параметрів в прямому порядку

info(price=5, object='горнятко', color='оранжевий') # передача параметрів у довільному порядку

# можна змішувати обидва способи, але спочатку розташовують параметри, які передають в прямому порядку

info('кава', price=10, color='чорний')
```

Приклад 4. Ця функція повертає аргумент, помножений на два, якщо він – від‘ємний, або аргумент, помножений на три, якщо він - більше або дорівнює нулю

```
def function(x):
     if x < 0:
          return x * 2
     else:
          return x * 3
def main():          # Виведення значень функції з діапазону [-3, 3]
     for i in range(-3, 4):
          y = function(i)
          print('function(', i, ') = ', y, sep='') # Виклик функції
main()
```

Приклад 5. Локальні та глобальні змінні

Приклад 5.1. Локальна змінна

```
def function():
    var = 'локальна змінна'      # визначення локальної змінної
    print(var)                   # виведення значення локальної змінної на екран
var = 'глобальна змінна' # визначення глобальної змінної
function()
print(var) # виведення на екран значення глобальної змінної
```

Приклад 5.2. Глобальна змінна

global *вказує на необхідність отримати доступ до глобальної змінної var, а не створювати нову локальну під час спроби що-небудь їй присвоїти*

```
def function():
    global var
    print(var) # виведення значення глобальної змінної на екран
    var = 'нове значення' # зміна глобальної змінної
    print(var) # виведення значення глобальної змінної на екран
var = 'глобальна змінна'
function()
print(var)
```

Приклад 5.3. Глобальні та локальні змінні

```
def function(c, d):          # a, b – глобальні змінні; c, d -- локальні
    global a, b
    a = 5; b = 7             # зміна значення глобальної змінної
    c = 10; d = 12 117       # зміна значення локальної змінної
a, b, c, d = 1, 2, 3, 4 # множинне присвоєння
print(a, b, c, d)
function(c, d)
print(a, b, c, d)
```

Приклад 6. **Рекурсивна функція**

Кожна з функцій може викликати інші функції, у тому числі звертатися до самої себе. Функцію, яка звертається до самої себе, називають ***рекурсивною.***

Приклад 6.1 ***Обчислення факторіалу***

а) ***рекурсивна функція*** обчислення факторіалу виглядатиме так:

```
def fact (num):
    if num == 0:
        return 1 # за домовленістю факторіал нуля дорівнює одиниці
    else:
        return num * fact (num - 1) # повертає добуток num і результату
                                      повернутого функцією fact (num - 1)
n=5
print("факторіал = ",fact(n))
```

б) обчислення факторіалу з ***використанням циклу***:

```
n = input("Факторіал числа ")
n = int(n)
fact = 1
i = 0
while i < n:
  i += 1
  fact = fact * i
print ("факторіал = ",fact)
```

Приклад 6.2 ***Рекурсивна функція обчислення суми чисел у списку***

Рекурсивний цикл закінчується і повертається нуль, коли функція отримає порожній список. Коли використовують рекурсію, то на кожному рівні рекурсії формується змінна L

а) пряма рекурсія:

```
def mysum(L):
    if not L:
        return 0
    else:
        return L[0] + mysum(L[1:]) # Викликає саму себе
mysum([1, 2, 3, 4, 5])
```

На кожному рівні рекурсії список стає все менше і менше, поки не спорожніє, що викличе кінець рекурсивного циклу. Сума обчислюється в процесі зворотного розкручування рекурсії.

Додайте у функцію оператор виведення L (print(L)) і запустіть приклад ще раз, щоб побачити вміст списку на кожному рівні рекурсії:

б) непряма рекурсія:

```
def mysum(L):
     if not L:
          return 0
     return nonempty(L) # Виклик функції, яка викличе цю функцію
def nonempty(L):
     return L[0] + mysum(L[1:]) # непряма рекурсія
mysum([1, 2, 3, 4, 5])
```

Індивідуальні практичні завдання

Завдання 1 **Напишіть програмну реалізацію наступного завдання:**

Створіь свою функцію my_max (), яка приймає два числа і повертає максимальне з них (насправді, така функція вже вбудована в Пітон :))

Програмний код помістіть у файл *lab_rob_3_6-1_**Прізвище.**py* та завантажте у віртуальне навчальне середовище.

Завдання 2 **Напишіть програмну реалізацію наступного завдання:**

Створіь свою функцію my_max (), яка приймає будь-яку кількість чисел і повертає максимальне з них

Програмний код помістіть у файл *lab_rob_3_6-2_**Прізвище.**py* та завантажте у віртуальне навчальне середовище.

Контрольні запитання

1. Що таке процедура та функція? Навіщо їх використовують?
2. Охарактеризуйте локальні та глобальні змінні. В чому їх відмінність?
3. Навіщо використовувати параметри під час визначення функції? Які типи параметрів існують?

Розділ ІІ. *JAVA*

Практичне заняття № 1
«Встановлення і налаштування середовища розробки мовою Java»

Мета: оволодіння базисними поняттями щодо встановлення і налаштування середовища розробки мовою Java.

Після практичного заняття студенти (курсанти) повинні:

вміти: встановити та налаштувати середовище розробки мовою Java.

знати: процедуру встановлення на ПК середовища розробки мовою Java.

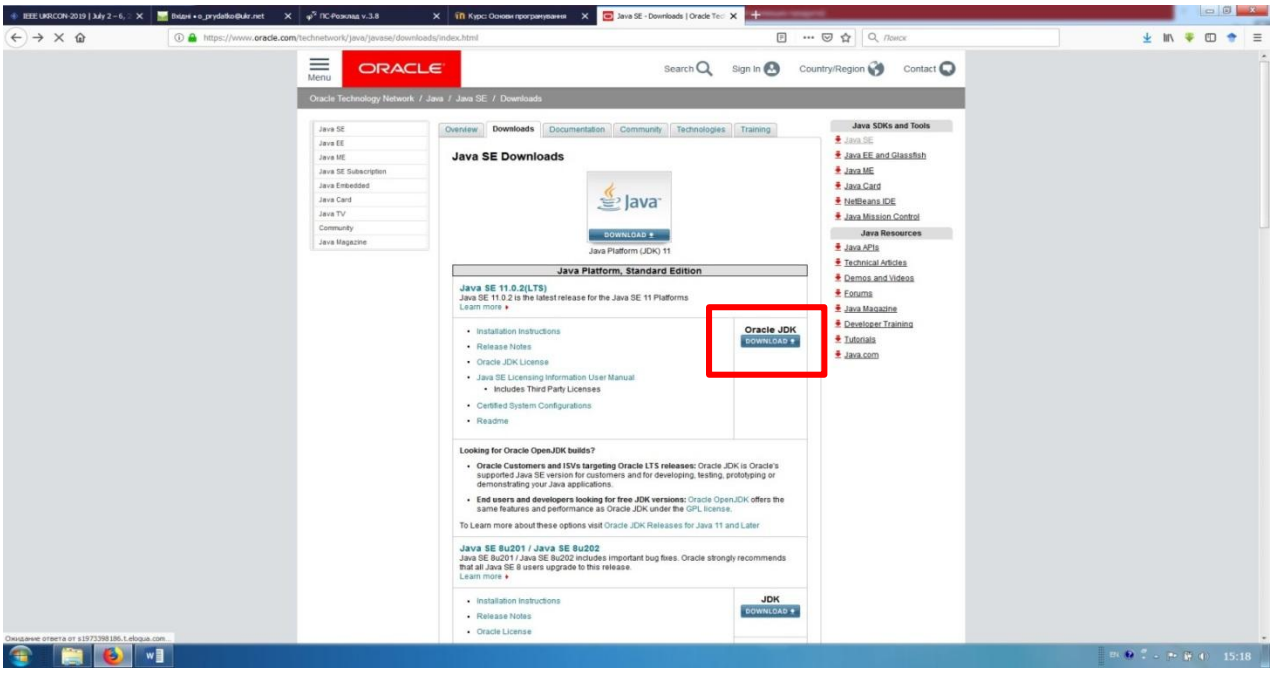
План заняття:

1. Завантаження установника:
2. Встановлення JDK на комп'ютері:

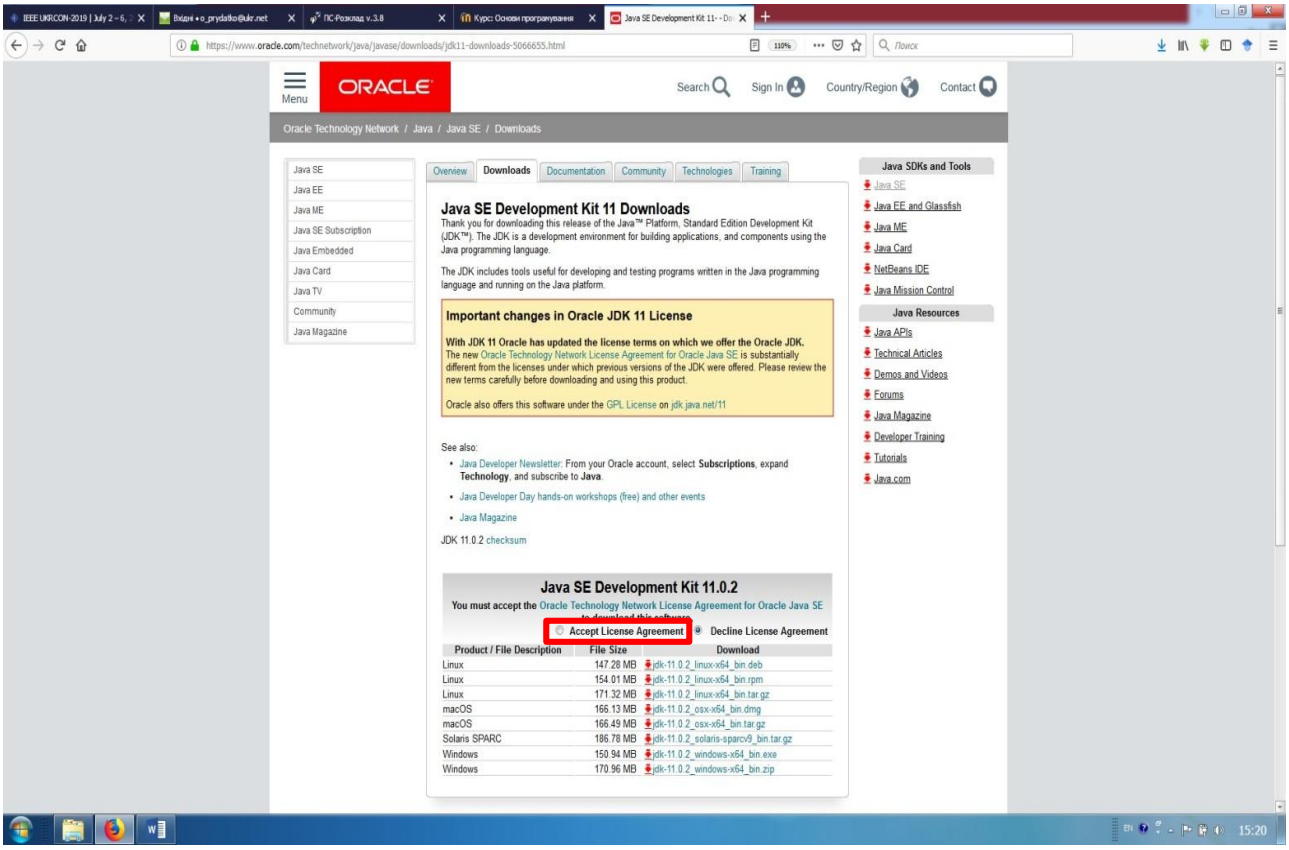
ВСТАНОВЛЕННЯ І НАЛАШТУВАННЯ СЕРЕДОВИЩА РОЗРОБКИ МОВОЮ JAVA

1. Встановлення та налаштування JDK.

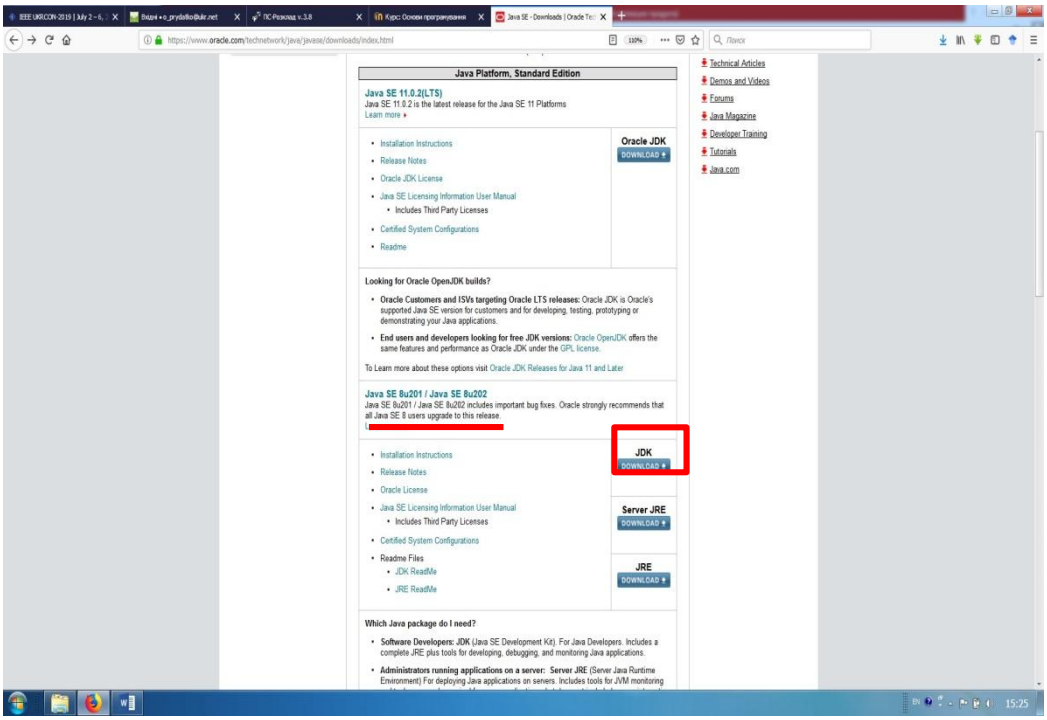
Як було зазначено на лекції, для програмування мовою Java, обов’язково потрібно встановити комплект розробника Java – JDK. Завантажити JDK можна з офіційного сайту компанії Oracle за посиланням: http://www.oracle.com/technetwork/java/javase/downloads/index.html. Або просто задавши в пошуковій системі «JDK».

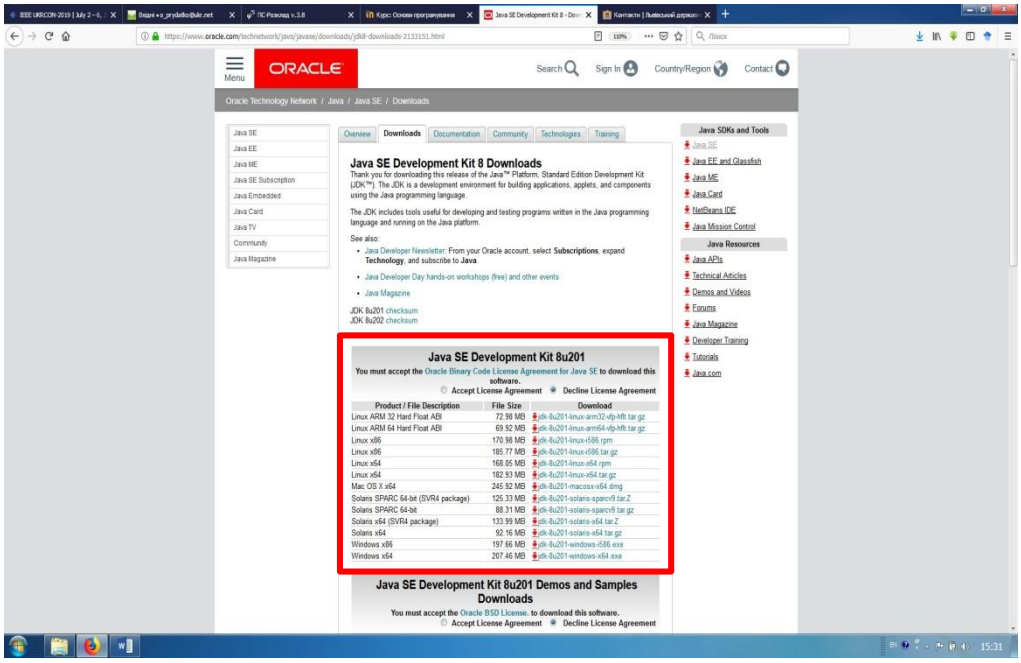


Пройшовши за адресою або посиланням пошукової системи, на офіційній сторінці корпорації Oracle відображатиметься посилання JDK Download з доступом до відповідних файлів. Після переходу за вказаним посиланням користувачеві буде за- пропоновано погодитись з умовами ліцензії (погодження обов'язкове – "Accept License Agreement").



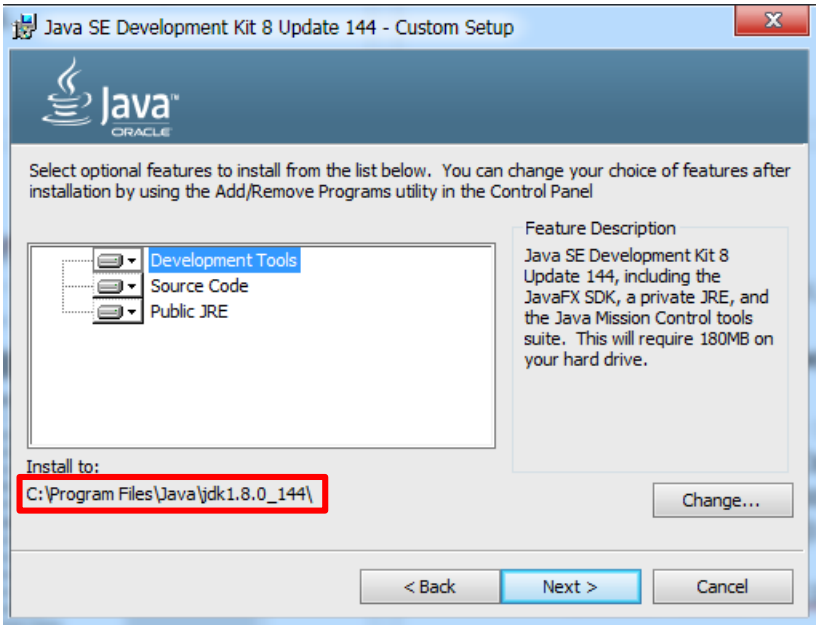
З представленого переліку необхідно обрати версію JSE Development Kit, яка підходить під операційну систему Вашого комп'ютера. На момент написання методичних рекомендацій, корпорація Oracle вже опублікувала 11 версію JDK, яка орієнтована лише на 64-розрядні операційні системи. Якщо існує необхідність встановлення комплекту розробника для 32-розрядної системи, то потрібно повернутись на попередню сторінку сайту, здійснити прокрутку нижче та вибрати версію JDK8, або в області пошуку ввести назву потрібної версії. Серед представленого переліку версій JDK8 можна підібрати необхідну.





Після завантаження інсталяційного файлу (це займе деякий час) необхідно йо- го інсталювати на ПК. Процес інсталяції не передбачає жодної надскладної операції, достатньо лише слідувати інструкціям інсталяційної програми.

Проте існує деякий нюанс. Під час налаштування процесу інсталяції буде автоматично запропоновано встановлення JDK в папку Program Files комп’ютера.



У випадку використання IDE, з метою подальшого програмування, таке місце розташування цілком підходить та не потребує заміни. Проте, якщо Ви націлені на програмування із використанням Notepad та командного рядка, то адресу інсталяції необхідно замінити на будь-який каталог за межами Program Files. Це пов'язано з тим, що для запуску компіляції програми з допомогою командного рядка необхідно задавати шлях доступу (місце розташування) до java-компілятора javac. А цей доступ з командного рядка в каталог Program Files може бути закрито (залежить від на- лаштування ОС). Відповідно, якщо Ви ставите перед собою мету спробувати програмувати через командний рядок, то інсталяцію JDK необхідно скерувати в каталог за межами Program Files. В нашому прикладі це буде підкаталог Java в кореневому каталозі C:\ (C:\Java\jdk1.8.0_144\).

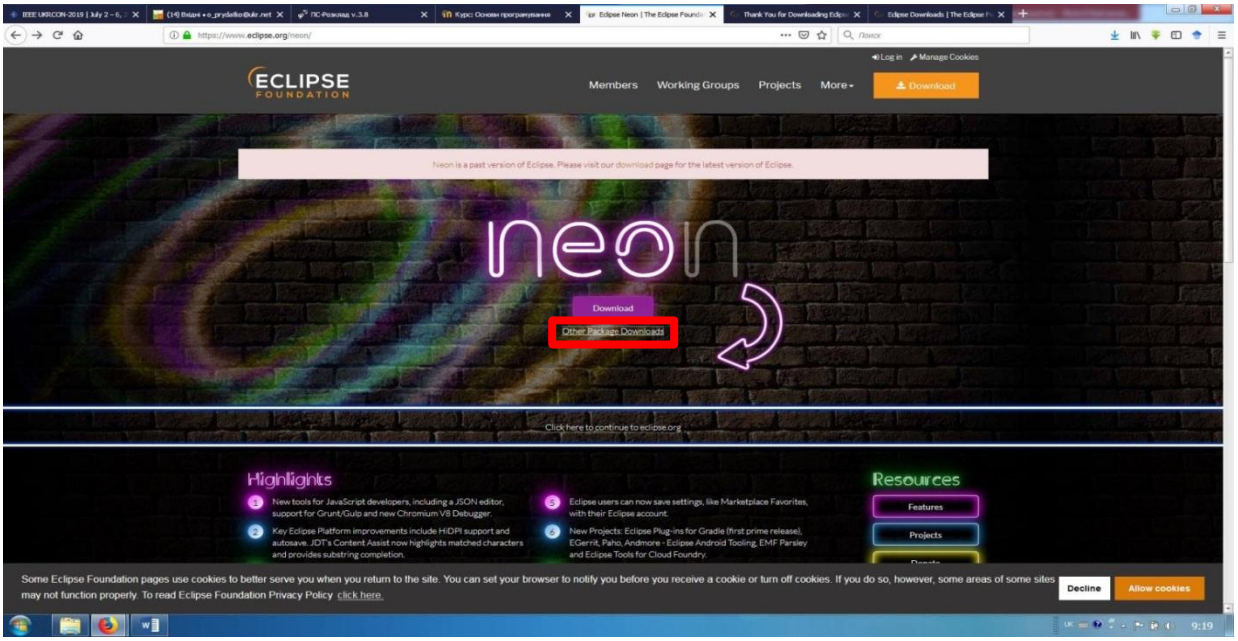
Важливо, що в процесі інсталяції також буде запропоновано замінити місце розташування JRE, то в цьому випадку адреси змінювати не потрібно!!!

2. Встановлення та налаштування IDE.

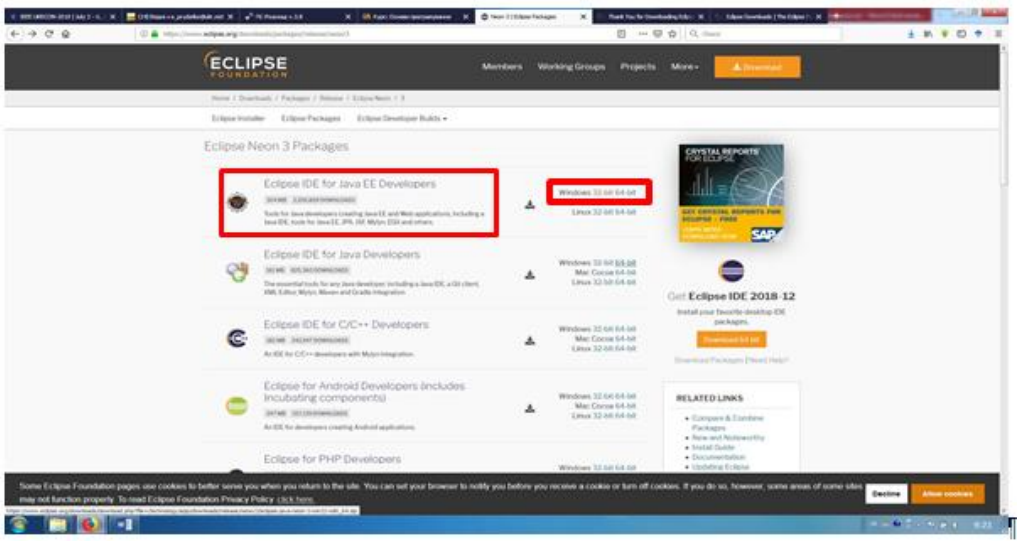
Наступним кроком для роботи з Java є ***встановлення IDE***. Для встановлення інтегрованого середовища розробки Eclipse відповідної версії (перелік останніх вер- сій Eclipse розглянуто в лекції) необхідно перейти за одним із посилань (офіційний сайт Eclipse):

http://www.eclipse.org/neon/ - версія Neon (червень 2016);
http://www.eclipse.org/oxygen/ - версія Oxygen (червень 2017);
http://www.eclipse.org/photon/ - версія Photon (червень 2018)…

Далі в прикладі описано процедуру інсталяції версії 2016 року Neon. Перейшовши за посиланням користувач потрапить на головну сторінку вказаної версії IDE.



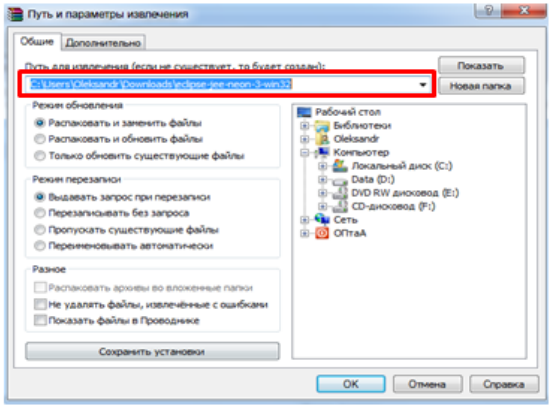
Перейшовши за посилання Download користувачеві буде надана можливість завантажити версію лише для 64-х розрядної операційної системи (останнє оновлення версії від 2018 року). У разі необхідності встановлення версії на 32-х розрядну систему необхідно перейти за посиланням Other Package Downloads та обрати необхідну версію.

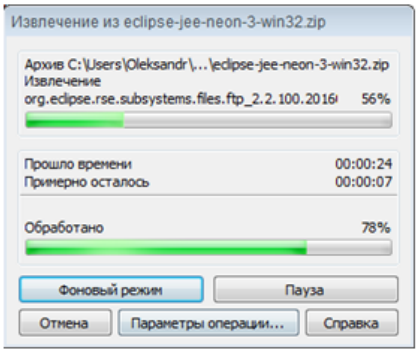


В зазначеному вікні користувачеві буде надана можливість обрати версію IDE для конкретної мови програмування (центральна частина) або різновиду операційної системи (права частина вікна). Для подальшого вивчення мови Java в межах курсу рекомендовано завантажити Eclipse **Neon** Packages → Eclipse IDE for Java EE Devel- opers.

Пакет середовища розробки завантажуватиметься у вигляді архіву. Після завантаження його необхідно розпакувати у зазначене користувачем місце. Адресу куди буде розпаковано пакет необхідно зазначити власноруч (рекомендовано), або запам’ятати адресу запропоновану за замовчуванням.

Це потрібно для того, щоб в подальшому знайти файл запуску *.exe.





Пройшовши за зазначеною адресою розпакованого архіву користувачеві надається доступ ***до файлу запуску середовища***. Не до файлу запуску інсталяції, а до файлу запуску середовища! Інсталювання середовища проводити не потрібно, розпакований пакет працюватиме як повноцінний додаток. Рекомендовано створити ярлик із запуском середовища на робочому столі або панелі швидкого запуску.

P.S. Поздоровляю! Середовища розробки мовою Java коректно встановлене і налаштовано. Приємного користування середовищем та успішних проектів!!!

Практичне заняття № 2
«Перший за стосунок в IDE Eclipse»

Мета: здобути навики щодо створення нового Java-проекту та написання найпростішої програми із виводу стрічки у консоль. Закріпити теоретичні знання щодо структури програмного коду на Java.

Після практичного заняття студенти (курсанти) повинні:

вміти: реалізовувати найпростіші програми із виводу текстових стрічок у консоль середовища розробки Eclipse.

знати: порядок створення нового Java-проекту та його структури, а також структури програмного коду написаного мовою Java.

План заняття:

1. Перший застосунок в середовищі розробки
2. Структура коду програми
3. Індивідуальне практичне завдання

Хід роботи

2.1. Перший застосунок в середовищі розробки

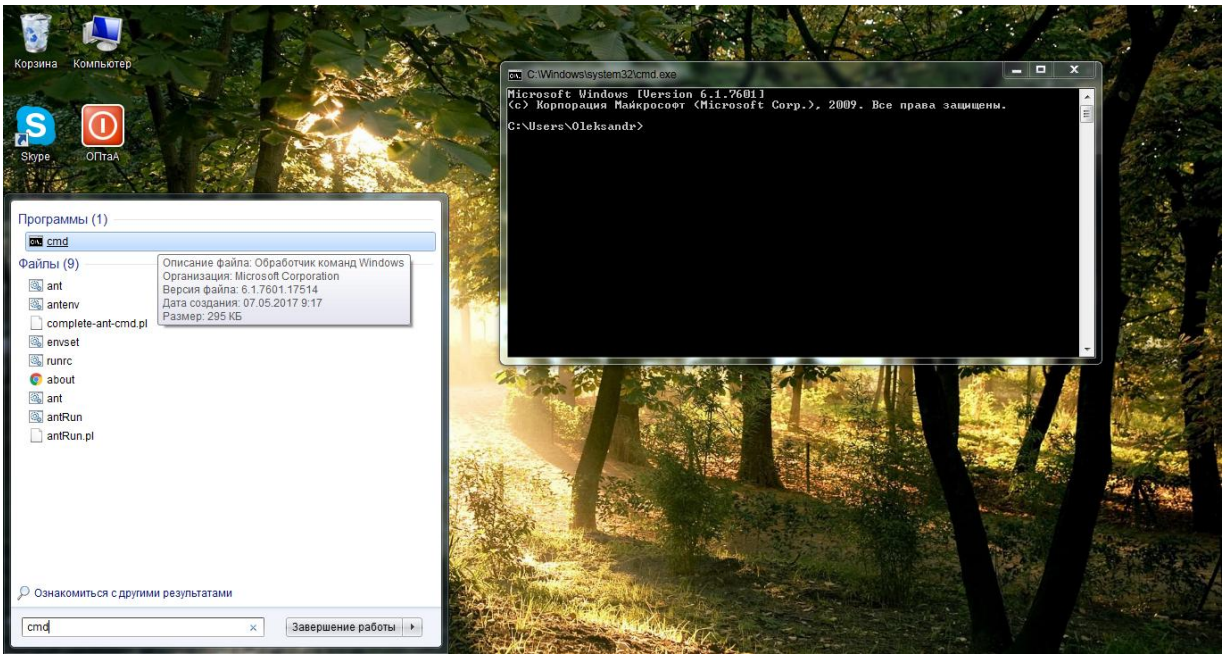
В якості першого застосунку реалізуємо найпростішу програму із виводу на екран стрічки «Hello world!». Спершу реалізуємо застосунок без використання IDE (з метою повноти уяви про трудомісткість процесу програмування без IDE).

Для написання коду програми запускаємо звичайний Notepad (як згадувалось в лекції) та набираємо код програми (про логіку коду пізніше):

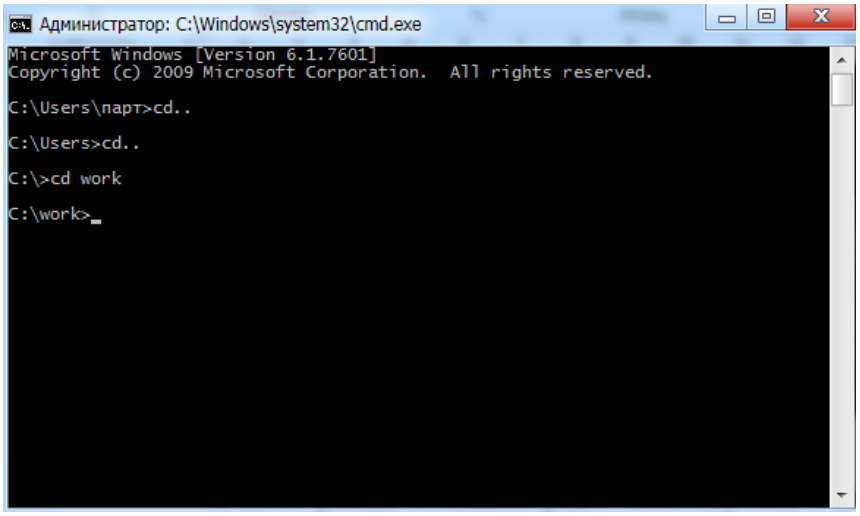
```
public class HelloWorld {
     public static void main(String[] args) {
          System.out.println("Hello world");
     }
}
```

Далі необхідно зберегти цей файл з розширенням .java та назвою, яка відповідає назві класу – HelloWorld. Місце збереження файлу не має значення, проте з метою полегшення процесу доступу до файлу, рекомендовано обрати місце збереження вище до кореневого каталогу. В нашому прикладі створено папку «work» в кореневому каталозі C:\.

Наступним кроком у виконанні програми є запуск командного рядка. Для виконання цієї операції необхідно у вікні пошуку програмного меню «Пуск» ввести «cmd».



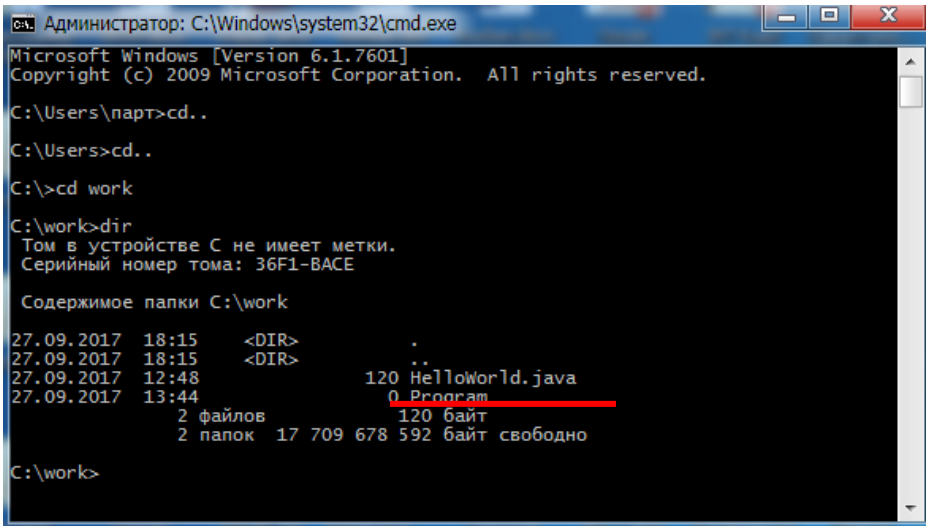
За замовчування командний рядок завантажується із доступом до папки користувача. З метою зміни адреси доступу (переходу до підкаталогу *вищого* рівня) необхідно виконати команду «cd..» потрібну кількість разів. Для переходу до підкаталогу *нижчого* рівня необхідно застосувати команду «cd» після якої вказувати ім’я наступного за ієрархією підкаталогу. В нашому прикладі потрібно здійснити перехід до підкаталогу «work» кореневого каталогу C:\, як це зображено на прикладі:



В папці «work» попередньо збережено файл з кодом програми HelloWorld.java. Переконатись в цьому можливо з допомогою командного рядка. З цією метою застосовують команду «dir»:

```
C:\work>dir
```

В результаті виконання команди буде виведено інформацію про відповідний підкаталог та розміщені у ньому файли:



Як видно з представленого прикладу в зазначеній папці міститься файл HelloWorld.java. Наступним кроком у виконанні програми є компіляція файлу з кодом програми.

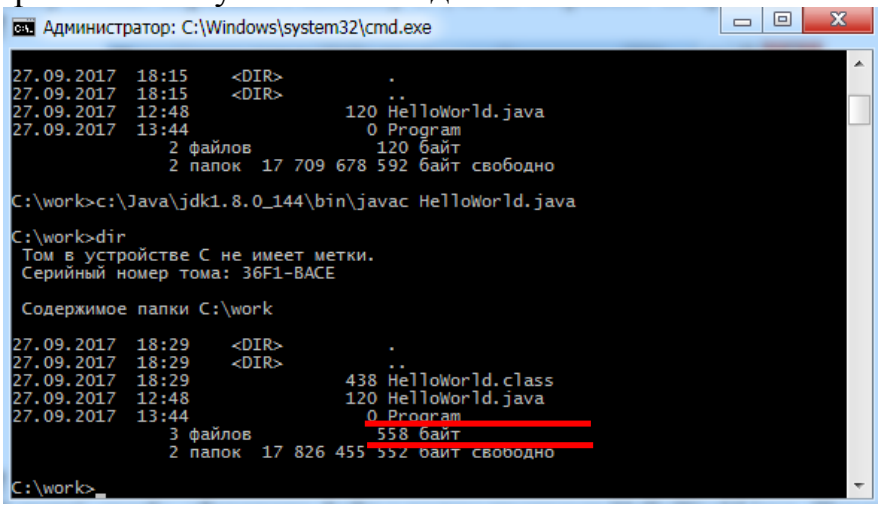
Для того, щоб зв'язати зазначений файл з компілятором java, необхідно вказати шлях доступу до javac. Місце розміщення компілятора відповідає директорії встановлення JDK, в зазначеному прикладі це:

```
C:\work>c:\Java\jdk1.8.0_144\bin\javac
```

Для виконання компіляції, після зазначення адреси розміщення javac, необхідно вказати ім'я файлу з розширенням за прикладом:

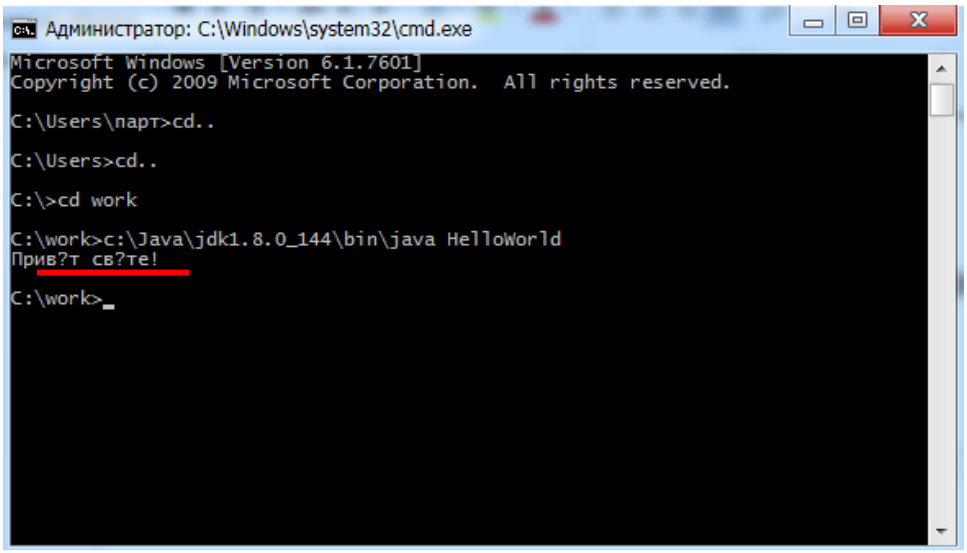
```
C:\work>c:\Java\jdk1.8.0_144\bin\javac HelloWorld.java
```

Після виконання компіляції, в командному рядку не відбудеться жодних змін, проте за адресою підкаталогу з вихідним кодом програми з'явиться ще один файл з ідентичною назвою та іншим розширенням (.class) – це файл з відкомпільованим байт-кодом програми. Переконатись в цьому можливо шляхом повторного застосування команди «dir»:



З метою виводу в командний рядок результатів компіляції необхідно виконати дії із зазначенням місця розташування JVM, яка виконуватиме відповідний байт-код:

```
C:\work>c:\Java\jdk1.8.0_144\bin\java HelloWorld
```



Результатом роботи програми буде вивід стрічки без відтворення деяких символів (або взагалі набір незрозумілих символів). Це результат виконання невідповідної кодової сторінки, для заміни якої необхідно повернутись до каталогу користувача, в нашому випадку це `C:\Users\парт` та використати команду:

```
C:\Users\парт>chcp 1251
```

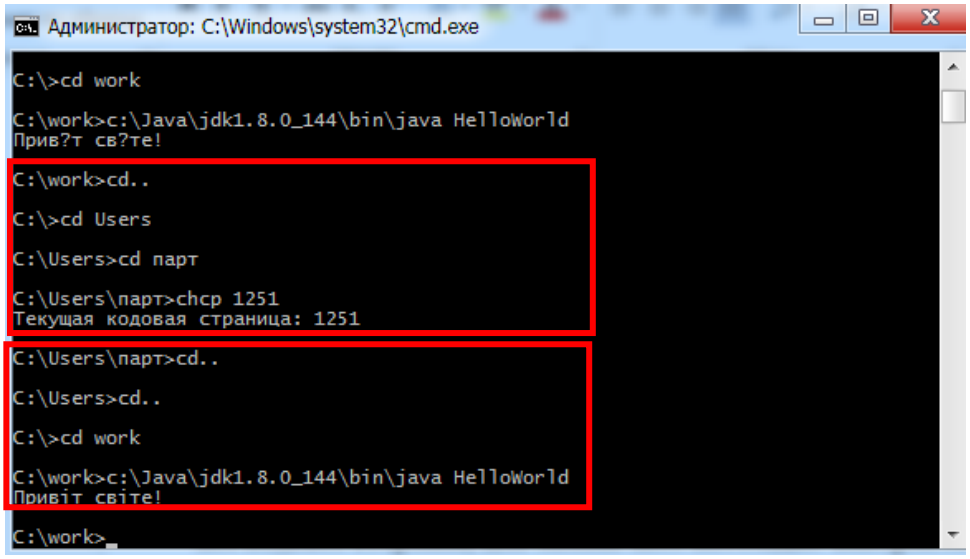
Або ж реалізувати виконання цієї команди перебуваючи безпосередньо в каталозі «work»:

```
C:\work>chcp 1251
```

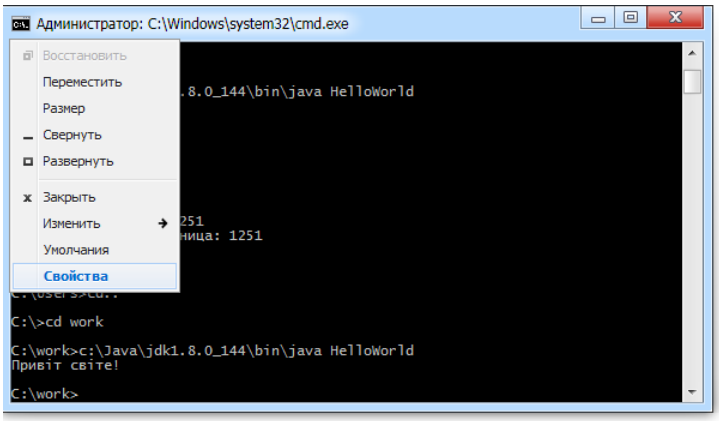
Після чого повернутись повторно виконати програму:

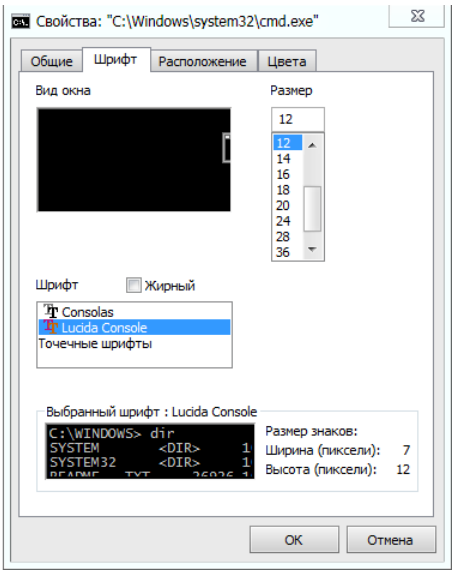
```
C:\work>c:\Java\jdk1.8.0_144\bin\java HelloWorld
```

Результатом роботи програми буде вивід стрічки «Привіт світе!»



Якщо в результаті роботи програми все ж таки одержана стрічка з незрозумілих символів, потрібно переконатись у відповідності шрифтів виводу тексту в консоль. Для цього необхідно перейти в меню командного рядка та обрати категорію «властивості». За обраною категорією в закладці «шрифт» необхідно обрати шрифт Lucida Console або Consolas.

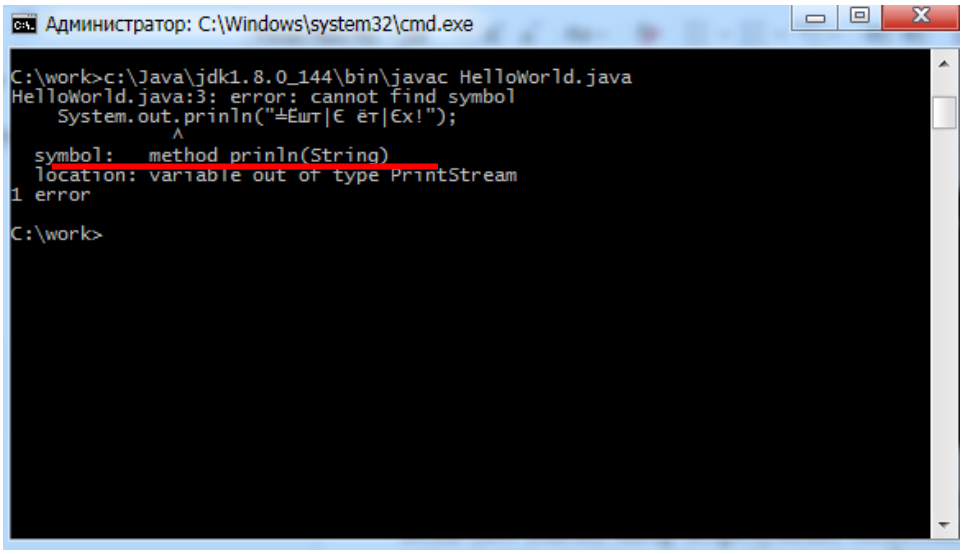




Слід зауважити, що в разі допущення в первинному коді програми будь-якої помилки, текстовий редактор цього не повідомить. Відстежити посилку можливо лише після компіляції програми з допомогою командного рядка. До прикладу, допустимо помилку в методі **println** :

```
public class HelloWorld {

    public static void main (String [] args){

        System.out.prinln("Привіт світе!");

    }

}
```

В результаті компіляції буде отримано:

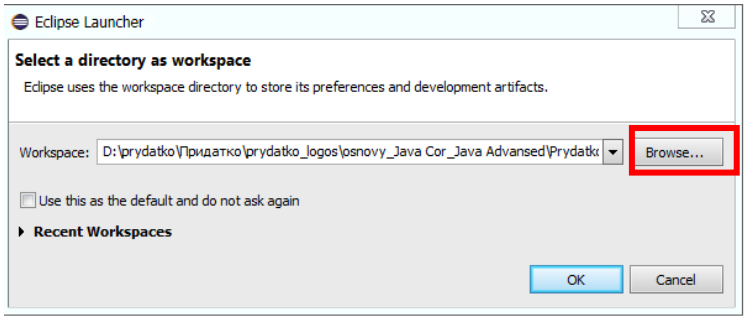


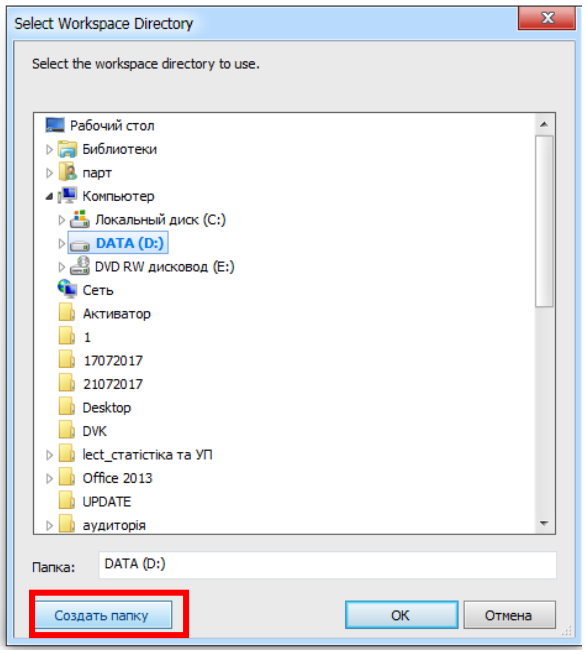
Для виправлення помилки необхідно повернутись до текстового редактора, виправити помилку та повторно запустити програму на компіляцію з допомогою командного рядка.

Власне незручність при роботі з помилками та трудомісткість процесу компіляції є основними недоліками такого методу програмування. З метою полегшення цього процесу використовують інтегровані середовища розробки, які орієнтовані на автоматичну перевірку синтаксису коду під час його написання та швидку компіляцію програми із виводом результату в консоль.

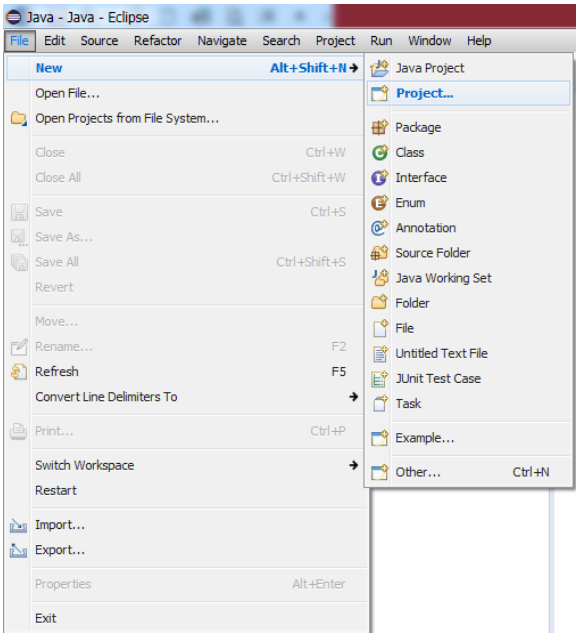
З метою наочності реалізуємо програму виводу стрічки «Привіт світе!» з використанням IDE.

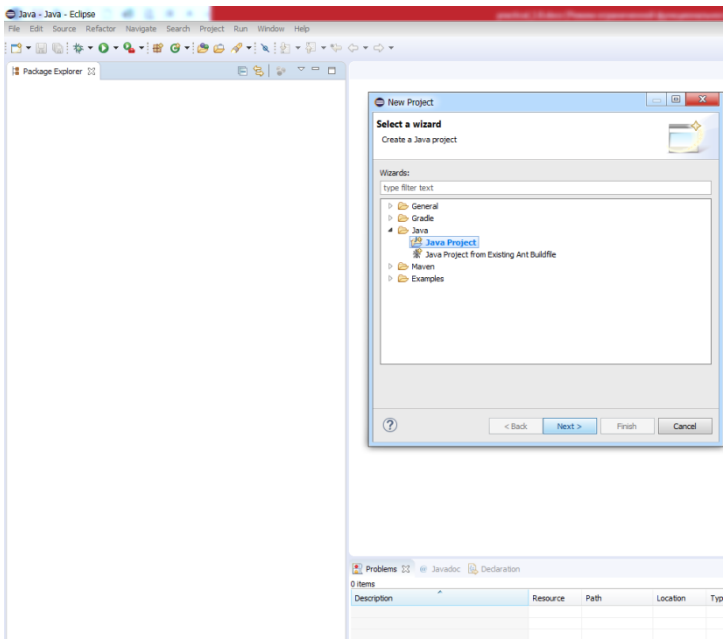
Під час запуску інтегрованого середовища розробки Eclipse необхідно зазначити каталог де зберігатимуться усі Java-проекти. Якщо середовище щойно встановлене та раніше не використовувалось, то таку директорію необхідно створити та вказати під час першого запуску середовища (створення каталогу доступне під час запуску IDE).



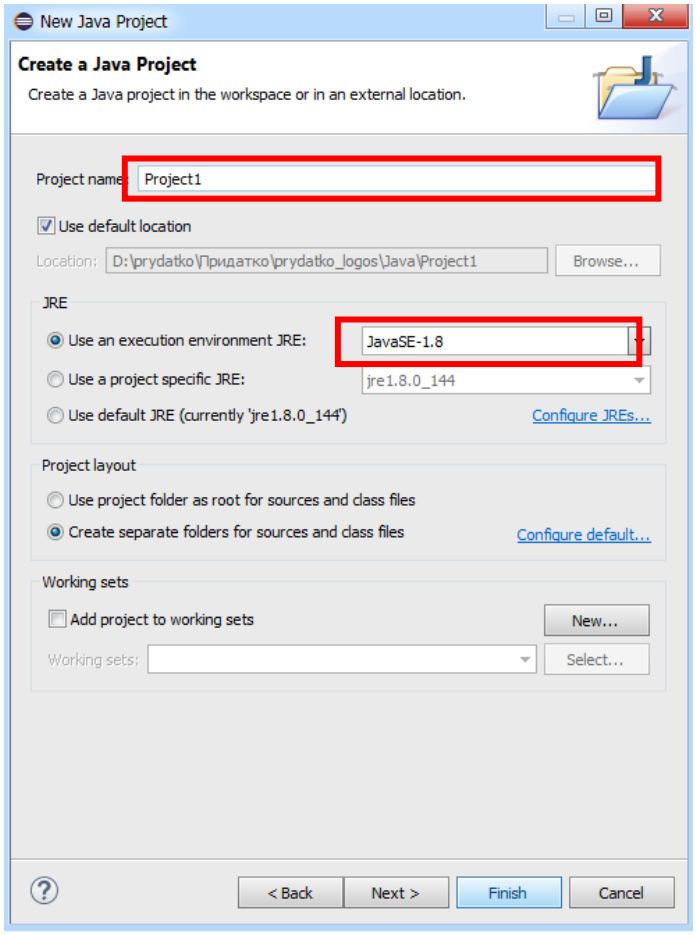


Після першого запуску Eclipse область Packege Explorer, звичайно, буде пустою. Для створення першого проекту необхідно перейти в меню File>New>Project. В вікні створення нового проекту необхідно вибрати Java>Java Project та пройти далі.

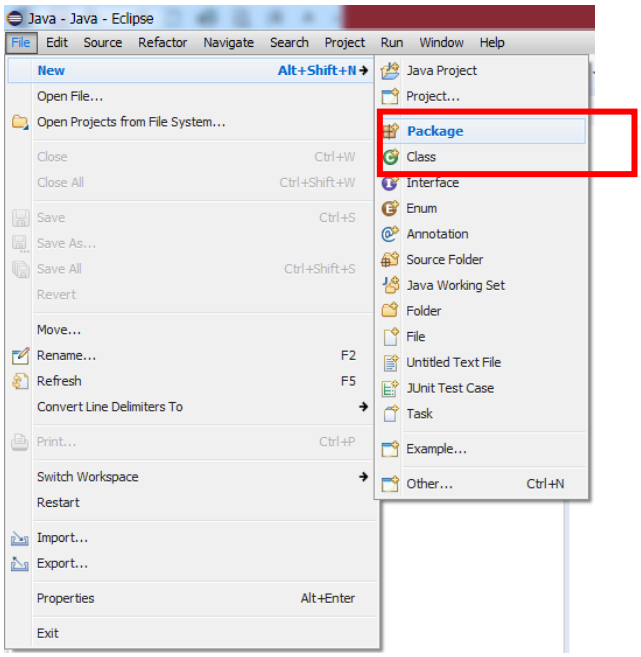




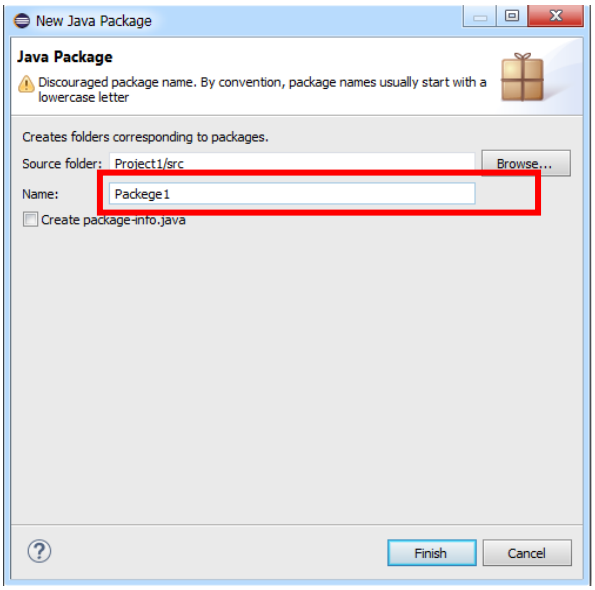
Наступним кроком є присвоєння імені для проекту. В цьому ж вікні можна вибрати версію Java, залежно від особливостей процесу розробки. За замовчуванням буде запропоновано останню версію JDK, яка встановлена на комп'ютері.



Java-проекти створюють з метою розміщення класів, проте деякі проекти можуть мати велику кількість класів (особливості об'єктно-орієнтованого програмування), тому з метою їх логічної структуризації в Java-проект введено елемент пакету. Для створення пакету необхідно знову звернутись до меню File>New>Packege (за умови виділеної папки з проектом) або натиснути на директорію пакету правою клавішею маніпулятора (миші) та виконати New>Packege. Аналогічним чином в межах пакету створюється клас. Нагадуємо, що клас являється основним файлом, що містить первинний код програми.

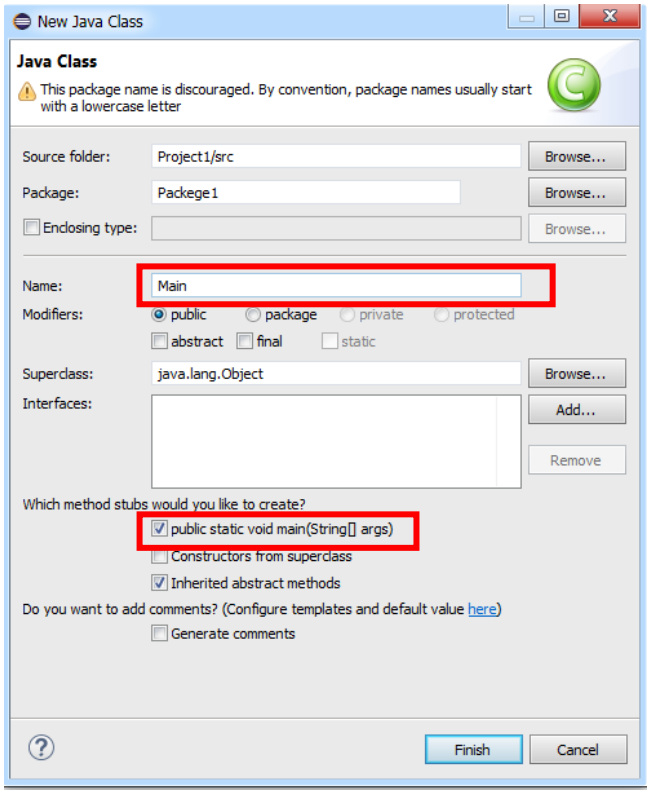


Після вибору позиції «створити новий проект», середовище запропонує дати назву новостворюваному пакету.



Новостворений пакет буде автоматично розміщений в каталозі «src». Слід зауважити, що в каталозі «src» буде збережено усю структуру Java-проекту. Крім того, при створенні нового проекту автоматично створиться бібліотека JRE System Library, яка потрібна для запуску програми.

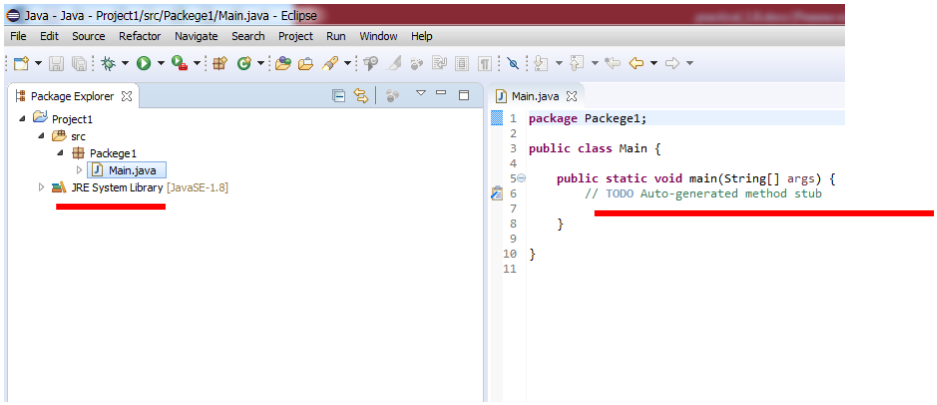
Дещо інший порядок створення класу. Виконавши аналогічні дії за умови виділеного новоствореного пакету (File>New>Class), середовище виведе вікно де необхідно зазначити назву класу. **Важливо!!!** Усі класи прийнято називати з великої літери. В прикладі це буде клас Main (головний). В зазначеному вікні користувачеві надається можливість встановити модифікатор доступу (про це пізніше), вибрати в якості місця збереження інший пакет, автоматично створити головний метод, генерувати конструктор або коментарі тощо. Для початку ми не будемо користуватись усіма представленими можливостями, окрім того, що встановимо позначку навпроти автоматичної генерації методу «public static void main (String [] args)» (щоб не набирати її потів власноруч).



Переваги використання IDE для створення класів очевидні. В результаті автоматичної генерації головного методу користувачеві не потрібно писати код

із назвою класу, оголошувати метод, слідкувати за синтаксисом, щоб не допустити помилки тощо. Усі ці операції Eclipse виконує автоматично.

У результаті проведених дій в Packege Explorer буде відображено новостворений метод, а в області написання коду – згенеровано головний метод:



В області написання коду Eclipse автоматично генерує стрічку:

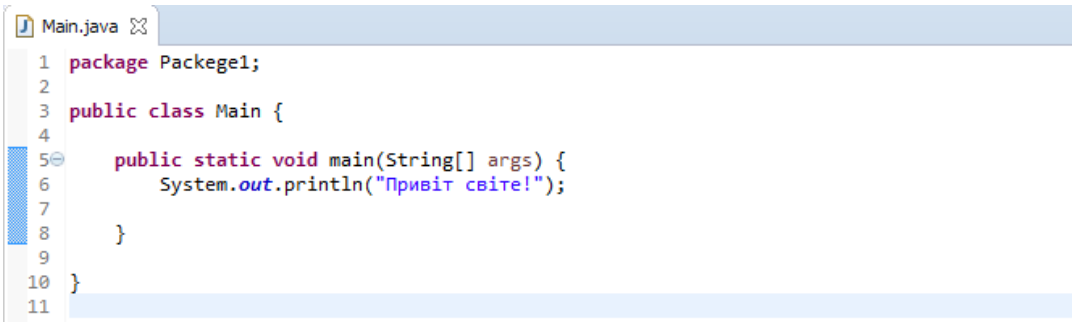
```
// TODO Auto-generated method stub
```

Це коментар, який не відіграє в програмі жодного значення окрім інформування про певні особливості того чи іншого методу. В нашому випадку цю стрічку можна видалити.

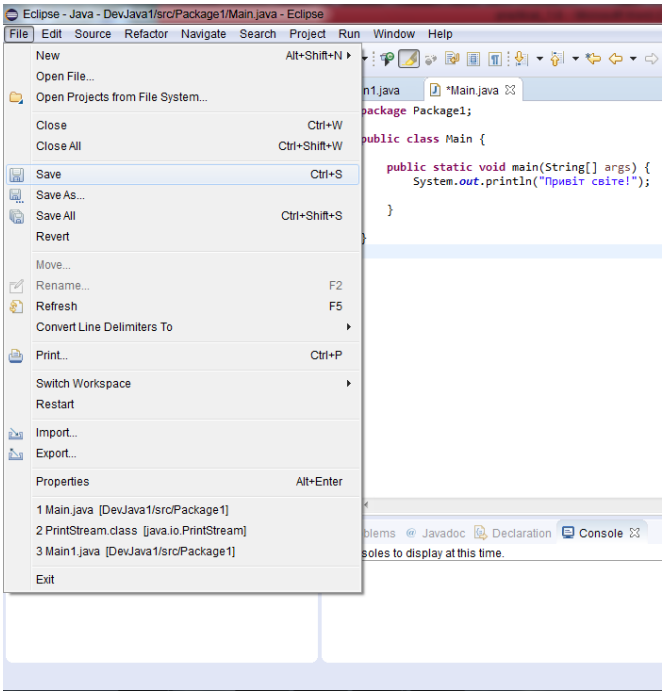
З метою формування власного коментаря необхідно в любій довільній області виділити необхідний текст та натиснути комбінацію клавіш Ctrl+/ (за умови англійської розкладки клавіатури).

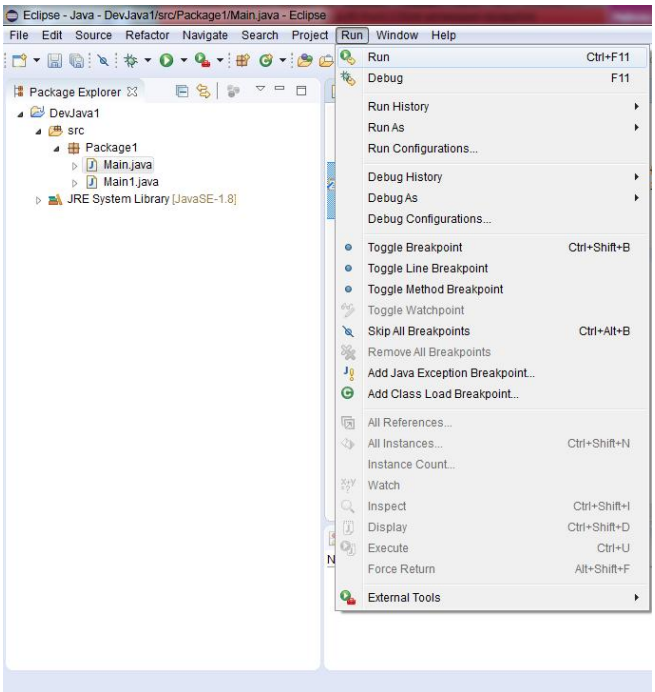
Якщо звернутись до первинного коду програми із виводу стрічки «Привіт світе!» (який було написано раніше), то можна відстежити, що у генерованому в Eclipse варіанті не вистачає лише (потрібно дописати):

```
System.out.println("Hello world");
```

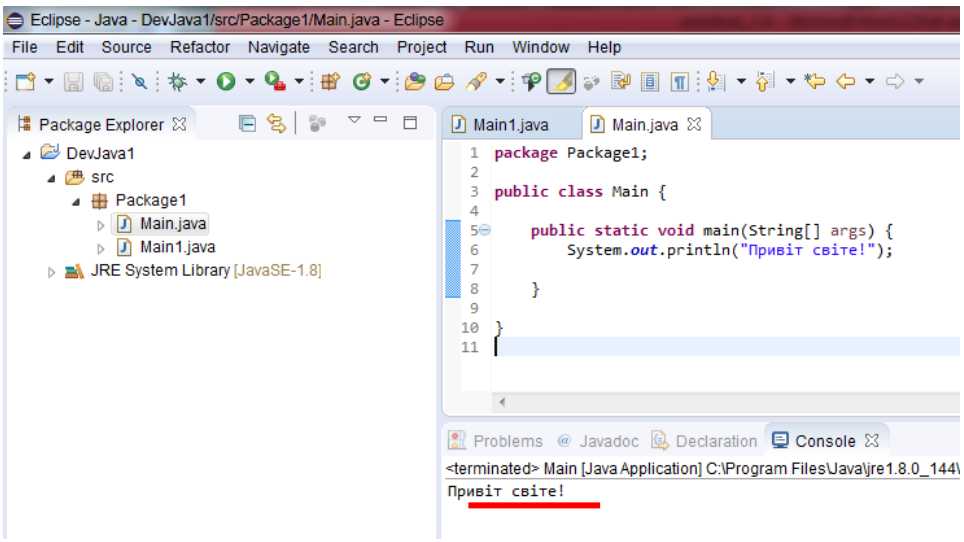


Завершивши написання коду необхідно зберегти напрацювання (File>Save або Ctrl+S) та запустити компілятор. Запуск коду на компіляцією із подальшим виконанням програми та виведенням результату в консоль здійснюється з допомогою натискання кнопки Run, меню Run>run, або комбінацією клавіш Ctrl+F11.

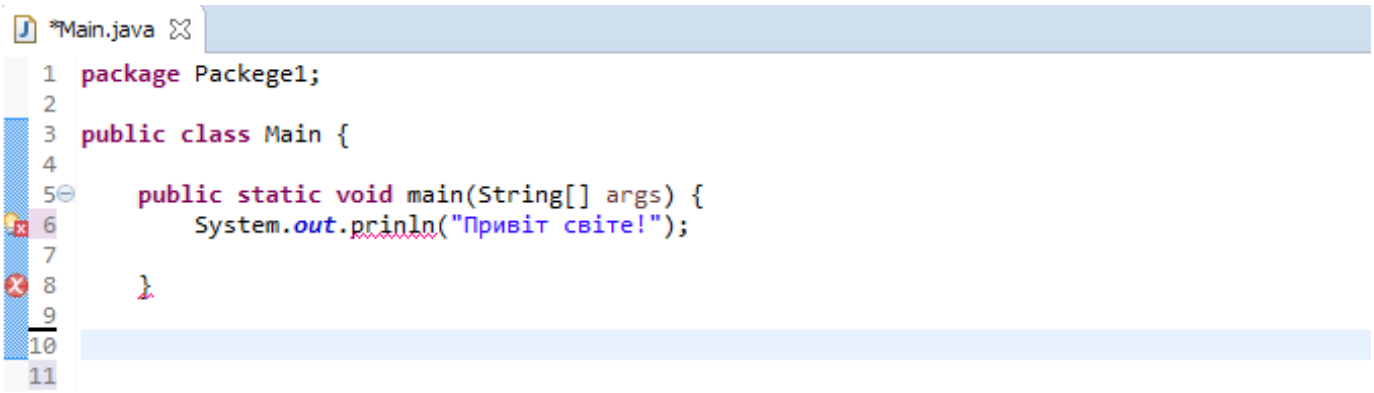




В результаті роботи програми в консоль буде виведено стрічку «Привіт світе!».

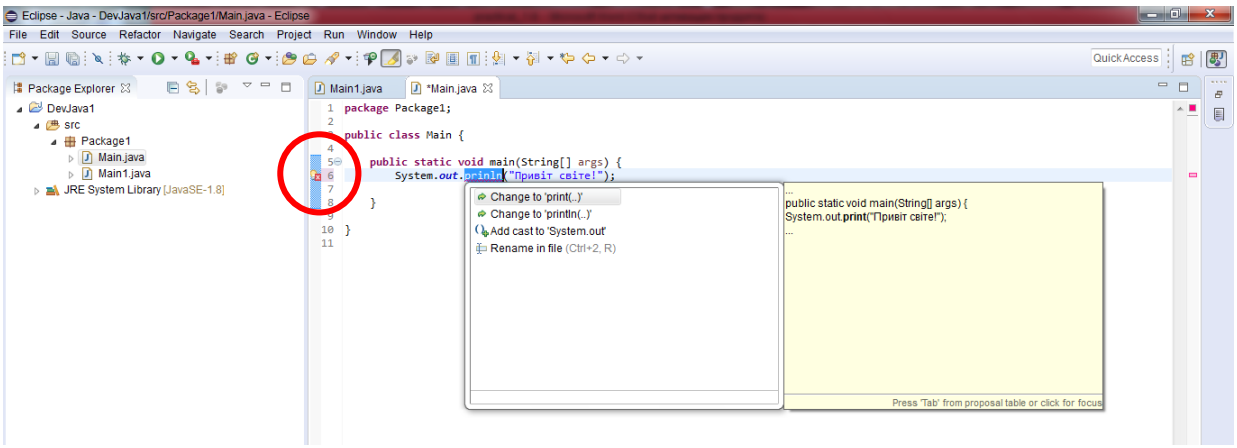


При допущенні помилки в процесі написання коду, компілятор автоматично повідомлятиме користувача про це:



Існує два типи помилок, синтаксична та логічна. Синтаксичні помилки пов’язані недотриманням правил написання коду, наприклад відсутня дужка, крапка з комою, лапки тощо. Логічні помилки це помилки у написання логіки коду, наприклад застосування не того методу, невірна ініціалізація змінної, або ж помилка у написанні методів (як зроблено в прикладі з методом println). В обох варіантах компілятор надає допомогу щодо можливих варіантів виправлення помилок.

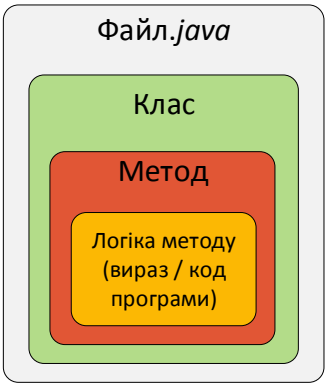
Для того щоб скористатись допомогою компілятора у виправлені помилки достатньо натиснути на піктограму, яка сигналізує про неї. В результаті чого середовище надає перелік можливих варіантів виправлення помилки у вигляді контекстного меню:



2.2. Структура коду програми

Для кращого розуміння ієрархії коду програми проведено його короткий аналіз. Для прикладу візьмемо вже добре відомий код програми із виводу стрічки «Привіт світе!»:

```
public class HelloWorld {

    public static void main (String []
    args){

        System.out.println("Привіт
        світе!");

    }

}
```



Для кращої уяви про структуру програмного коду поділимо його на три категорії: клас; метод; вираз. Усі з зазначених категорій виділені різними кольорами та відповідають наведеним прикладам.

Файл з первинним кодом (розширення java) містить в собі клас. Власне клас це і є програма, або її частина. Маленькі програми можуть складатись з одного класу. Усе, що віднесено до класу розміщується між фігурними дужками (так званий блок класу):

```
public class HelloWorld {

    public static void main (String [] args){

        System.out.println("Привіт світе!");

    }

}
```

Клас може вміщувати один або декілька методів. Власне методи містять в собі усі інструкції щодо виконання програми (код програми). Методи обов'язково потрібно розміщувати в середині класів (між фігурними дужками класу).

```
public class HelloWorld {
     public static void main (String [] args){
          System.out.println("Привіт світе!");
     }
}
```

Логіка методу (інструкції щодо виконання програми) має розміщуватись в середині методу. Іншими словами, код методу – це набір його виразів (що та як він робить).

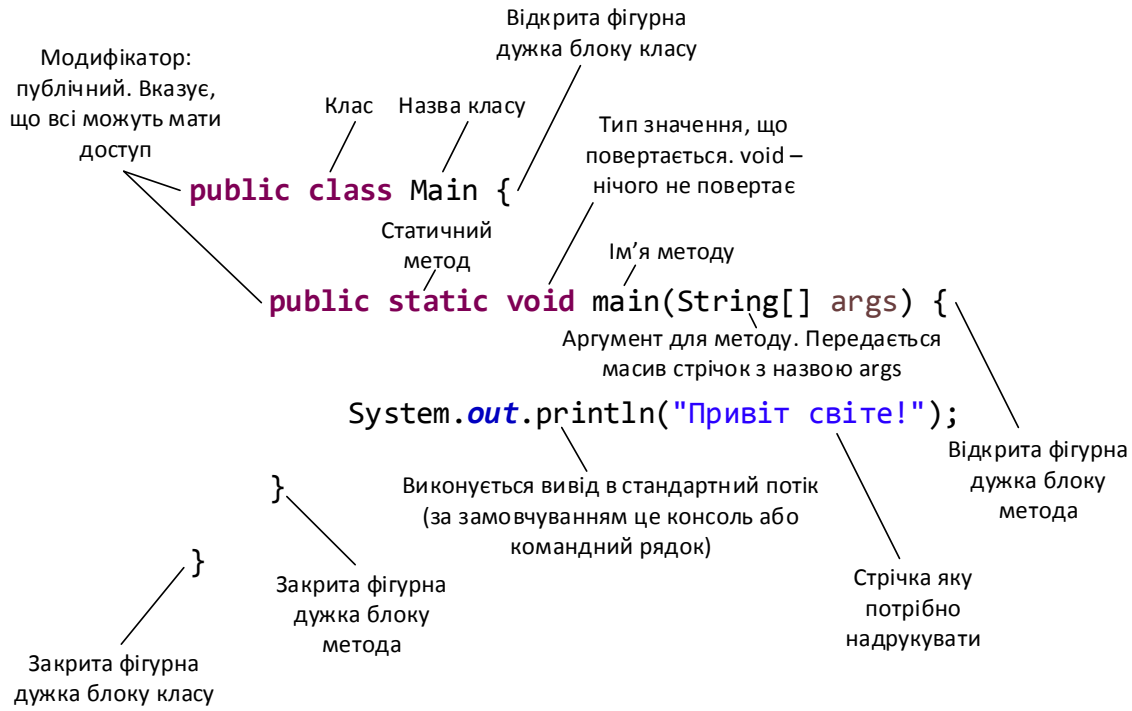
```
public class HelloWorld {
     public static void main (String [] args){
          System.out.println("Привіт світе!");
     }
}
```

В наведеному прикладі вираз методу виводить у консоль стрічку «Привіт світе!».

Отже, узагальнимо ієрархію коду: ***клас → метод → вираз (логіка програми).***

Коли JVM (віртуальна машина Java) починає свою роботу, вона першою чергою шукає клас (власне чому клас в ієрархії коду займає перше місце). Після знаходження класу JVM шукає метод та заходить в нього. Далі JVM почерзі виконує все, що знаходиться між фігурними дужками головного методу. Будь-яка програма написана на Java має що найменше один клас та головний метод.

Тепер зоглянемо детальніше кожен елемент представленого коду:



Не потрібно намагатись відразу запам'ятати всю структуру коду та вивчити, що як називається. Основна мета – це *ознайомлення* з елементами коду. Далі в процесі вивчення основ програмування користувачі неодноразово звертатимуться до структури та ієрархії коду, та більш детально ознайомляться з ним.

2.3. Індивідуальне практичне завдання

Завдання полягає у написанні найпростішої програми із виводу інформації про свою родину в такому форматі:

```
Батько: Чумаченко Степан Дмитрович, 1965 року народження, підприємець;
… …
Сестра: Чумаченко Надія Степанівна, 1998 року народження, студентка;
… …
```

Первинний код програми зберігається у файлі з розширенням .java за адресою яку вказано під час запуску середовища розробки в каталозі «src»:

…\назва каталогу з Java-проектом\назва проекту\src\назва пакету**файл.java**

Практичне заняття № 3
«Написання лінійних програм. Застосування операторів»

Мета: здобути навики щодо роботи з основними типами даних та операторами в мові програмування Java..

Після практичного заняття студенти (курсанти) повинні:

вміти: створювати елементарні програми із лінійним сценарієм виконання

знати: основні типи даних та особливості застосування основних операторів

План заняття:

1. Оголошення та ініціалізація змінних різних типів та застосування основних операторів
2. Індивідуальне практичне завдання

1. Оголошення та ініціалізація змінних різних типів

Для реалізації перших застосунків використовуючи основні типи даних розглянемо декілька розповсюджених прикладів.

Перед початком реалізації програми, в середовищі IDE Eclipse необхідно створити новий клас Arithmetic із автоматичною генерацією main методу (головного методу).

Приклад 1: Оголошення змінної типу int, ініціалізація її значенням та виведення у консоль:

```
public static void main(String[] arg) {
    int counter;
    counter = 99;
    System.out.println(counter);
}
```

Спростивши синтаксис зазначеного виразу можемо отримати:

```
public static void main(String[] arg) {
    int counter = 99;
    System.out.println(counter);
}
```

Результат роботи:
99

Приклад 2: Застосування основних арифметичних операцій. В ході написання зазначеного прикладу рекомендовано почергово виводити у консоль результат виконання арифметичних операцій:

```
public static void main(String[] arg) {
	int a;
	a = 23 + 87;
	System.out.println("Рузультат : "+a);
	int b = a - 44;
	System.out.println("Рузультат : "+b);
	int c = a * b;
	System.out.println("Рузультат : "+c);
	double d = c / 53;
	System.out.println("Рузультат : "+d);
	int e = c / 53;
	System.out.println("Рузультат : "+e);
	int f = b % 10;
	System.out.println("Рузультат : "+f);
	double g = (b + a) / (a * 11.5);
	System.out.println("Рузультат : "+g);
}
```

Результат роботи:

```
Рузультат : 110
Рузультат : 66
Рузультат : 7260
Рузультат : 136.0
Рузультат : 136
Рузультат : 6
Рузультат : 0.1391304347826087
```

Приклад 3: Застосування операторів з присвоєнням та посилкового типу даних String:

```
public static void main(String[] arg) {
	int x = 0;
	String s = "Значення x рівне: ";
	x += 20;
	System.out.println(s+x);
	x -= 15;
	System.out.println(s+x);
	x *= 2;
	System.out.println(s+x);
}
```

Результат роботи:

```
Значення x рівне: 20
Значення x рівне: 5
Значення x рівне: 10
```

Приклад 4: Застосування операторів інкремента та дектемента. Спочатку постфіксні оператори (зверніть увагу на результат виконання програми):

```
public static void main(String[] arg) {
   int x, a;
   x = 10;
   a = x++;
   System.out.println("x=" + x);
   System.out.println("a=" + a);
   a = x--;
   System.out.println("x=" + x);
   System.out.println("a=" + a);
}
```

Результат роботи:

```
x=11
a=10
x=10
a=11
```

Аналогічні операції з префісними операторами (зверніть увагу на результат виконання програми):

```
public static void main(String[] arg) {
   int x, a;
   x = 10;
   a = ++x;
   System.out.println("x=" + x);
   System.out.println("a=" + a);
   a = --x;
   System.out.println("x=" + x);
   System.out.println("a=" + a);
}
```

Результат роботи:

```
x=11
a=11
x=10
a=10
```

Ще один цікавий приклад з інкрементами та декрементами:

```
public static void main(String[] arg) {
     int b = 5;
     int a = 12;
     int c = ++a - b++;
     System.out.println(c);
     int d = b++ * 2;
     System.out.println(d);
}
```

Результат роботи:

```
8
12
```

Приклад 5: Особливості роботи з типом даних char:

```
public static void main(String[] arg) {
	char a = 'Л';
	char b = 'Д';
	char c = 'У';
	char d = 'Б';
	char e = 'Ж';
	char f = 'Д';
	System.out.print(a);
	System.out.print(b);
	System.out.print(c);
	System.out.print(d);
	System.out.print(e);
	System.out.print(f);
}
```

Результат роботи:

```
ЛДУБЖД
```

Приклад 6: Застосування логічних типів даних та їх операторів:

```
public static void main(String[] arg) {
	boolean a1 = true;
	boolean a2 = false;
	boolean a3 = a1 || a2;
	System.out.println(a3);

	boolean b1 = true;
	boolean b2 = false;
	boolean b3 = b1 && b2;
	System.out.println(b3);
	boolean c = true;
	boolean d = false;
	boolean e = true;
	boolean result = c && (d || e);
	System.out.println(result);

	boolean g = true;
	boolean h = !g;
	System.out.println(h);
}
```

Результат роботи:

```
true
false
true
false
```

Приклад 7: Застосування коментарів:

```
public static void main(String[] arg) {

      /*
      Приклад багаторядкового коментаря
      Застосування логічних операторів
      */

   boolean c = true; // Приклад однорядкового коментаря
   boolean d = false;
   boolean e = true;
   boolean result = c && (d || e);
   System.out.println(result); // виведення результату у консоль

}
```

2. Індивідуальні практичне завдання

Вкінці практичного заняття студенти (курсанти) отримують індивідуальне практичне завдання, результат виконанням якого необхідно завантажити в віртуальне навчальне середовище. Індивідуальне практичне завдання полягає у створенні простих програмних додатків із використанням основних операторів мови Java.

Завдання 1. Створити програму з двома змінними типу int таким чином, щоб вона виводила у консоль результат їх ділення та залишку від ділення.

Завдання 2. Створити програму, яка виводитиме у консоль суму цифр двозначного числа записаного до змінної типу int.

Завдання 3. Створити програму, яка виводитиме у консоль округлене число, записане до змінної типу double, до найближчого цілого.

Практичне заняття № 4
«Написання програм з розгалуженням»

Мета: здобути навики щодо застосування операторів вибору if та switch в процесі створення найпростіших програм.

Після практичного заняття студенти (курсанти) повинні:

вміти: створювати елементарні програми із розгалуженням логіки виконання коду використовуючи оператори вибору if та switch.

знати: принцип роботи та особливості застосування операторів вибору if та switch в програмуванні.

План заняття:

1. Написання програм з розгалуженням використовуючи умовний оператор if
2. Написання програм з розгалуженням використовуючи оператор множинного вибору switch
3. Індивідуальне практичне завдання

1. Написання програм з розгалуженням використовуючи умовний оператор if

Для реалізації перших застосунків використовуючи умовний оператор if розглянемо декілька розповсюджених прикладів.

Приклад 1: Створити програму, яка здійснює перевірку відношення записаного у змінну *n* цілого числа до парних або непарних чисел із виводом результату у консоль.

Для написання подібної програми слід згадати оператор «залишок ділення по модулю» та використати поєднання операторів if/else:

```
public class Main1 {

	public static void main(String[] args) {
		int n = 5;
		if (n%2==0){
			System.out.println("Змінна парна");
		}else{
			System.out.println("Змінна не парна");
		}
	}
}
```

Можуть зустрічатись випадки, коли в змінну *n* буде записано 0. Такий варіант програми матиме розгалуження if-else-if:

```
public class Main1 {

	public static void main(String[] args) {
		int n = 5;
		if (n==0){
			System.out.println("Введіть ціле цисло!");
		}else if (n%2==0){
			System.out.println("Змінна парна");
		}else{
			System.out.println("Змінна не парна");
		}
	}
}
```

Приклад 2: Створити програму, яка здійснює перевірку та виводитиме у консоль найближче до 10 з двох чисел записаних у змінні *i* та *j*.

Для написання подібної програми, крім використання операторів if-else-if необхідно ввести додаткові змінні та звернутись до методу «модуль числа» класу Math (Math.abs):

```
public class Main2 {

	public static void main(String[] args) {
		double i = 8.5;
		double j = 11.45;
		double n = 10;
		double m = (Math.abs(n-i));
		double k = (Math.abs(n-j));
		if (m>k){
			System.out.println(j);
		}else if (k>m){
			System.out.println(i);
		}else{
			System.out.println("Ви вказали однакові числа");
		}
	}
}
```

Приклад 3: Створити програму визначення та виводу у консоль дійсних коренів квадратного рівняння $ax^2+bx+c=0$, або повідомлення про те, що коренів немає. В змінні *a, b, c* мають цілочисельний тип даних.

Для написання подібної програми необхідно згадати шкільний курс математики та знову звернутись до класу Math із використанням методу «квадратний корінь числа» (Math.sqrt):

```
public class Main3 {
	public static void main(String[] args) {
		int a = 1;
		int b = 1;
		int c = 1;
		int D = (b*b)-(4*a*c);
		double x1;
		double x2;
		double x;
		System.out.println(D);
		if (D>0){
			x1 = (-b-(Math.sqrt(D)))/(2*a);
			x2 = (-b+(Math.sqrt(D)))/(2*a);
			System.out.println(x1+" "+x2);
		}else if (D==0){
			x = (-b)/(2*a);
			System.out.println(x);
		}else{
			System.out.println("Кореня немає");
		}
	}
}
```

2. Написання програм з розгалуженням використовуючи оператор множинного вибору switch

Для реалізації перших застосунків використовуючи оператор swich також розглянемо декілька елементарних прикладів.

Приклад 1: В змінну char записано оцінку за проходження тестування *без врахування регістру*. Створити програму, яка виводитиме у консоль повідомлення із привітанням або рекомендаціями щодо проходження тесту за таких умов: A – «Вітаємо, тест складено на відмінно!»; B – «Тест складено дуже добре!»; C – «Тест складно добре, слід звернути увагу на слабкі сторони»; D – «Тест складено задовільно, слід повторно переглянути матеріал»; E – «Тест ледве складено, рекомендовано повторно пройти навчання»; F – «Тест не складено!».

Для написання програми слід продумати, яким чином враховуватиметься регістр літер під час порівняння виразу switch із значеннями case:

```
public class Main1 {
	public static void main(String[] args) {
		char grade = 'c';
		switch (grade) {
		case 'A':
		case 'a':
```

```
                    System.out.println("Вітаємо, тест складено на
відмінно!");
                    break;
               case 'B':
               case 'b':
                    System.out.println("Тест складено дуже добре!");
                    break;
               case 'C':
               case 'c':
                    System.out.println("Тест складно добре, слід
звернути увагу на слабкі сторони");
                    break;
               case 'D':
               case 'd':
                    System.out.println("Тест складено задовільно, слід
повторно переглянути матеріал");
                    break;
               case 'E':
               case 'e':
                    System.out.println("Тест ледве складено,
рекомендовано повторно пройти навчання");
                    break;
               case 'F':
               case 'f':
                    System.out.println("Тест не складено!");
                    break;
               default:
                    System.out.println("Невірна оцінка");
               }
          }
     }
```

Приклад 2: Створити програму, яка відтворюватиме принцип роботи пульта керування ліфтом 9-ти поверхового будинку та здійснюватиме підйом або спуск із виводом результату у консоль «Ви піднялись на 6 поверх», або «Ви спустились на 1 поверх». Для ускладнення завдання))) кнопка 2-го поверху не працює! При її натисканні під час підйому ліфт здійснюватиме доставку на 3-й поверх, і навпаки, під час спуску – на 1-й.

Для реалізації цієї програми слід згадати про вкладений оператор switch:

```
public class Main1 {
     public static void main(String[] args) {
          String direction = "До низу";
          int floor = 6;
          switch (direction) {
```

```
            case "До гори":
                  switch (floor) {
                  case 1:
                        System.out.println("Ви піднялись на 1 поверх");
                        break;
                  case 2:
                  case 3:
                        System.out.println("Ви піднялись на 3 поверх");
                        break;
                  case 4:
                        System.out.println("Ви піднялись на 4 поверх");
                        break;
                  case 5:
                        System.out.println("Ви піднялись на 5 поверх");
                        break;
                  case 6:
                        System.out.println("Ви піднялись на 6 поверх");
                        break;
                  case 7:
                        System.out.println("Ви піднялись на 7 поверх");
                        break;
                  case 8:
                        System.out.println("Ви піднялись на 8 поверх");
                        break;
                  case 9:
                        System.out.println("Ви піднялись на 9 поверх");
                        break;
                  default:
                        System.out.println("Ви вказали неіснуючий поверх!");
                  }
                  break;
            case "До низу":
                  switch (floor) {
                  case 9:
                        System.out.println("Ви спустились на 9 поверх");
                        break;
                  case 8:
                        System.out.println("Ви спустились на 8 поверх");
                        break;
                  case 7:
                        System.out.println("Ви спустились на 7 поверх");
                        break;
                  case 6:
                        System.out.println("Ви спустились на 6 поверх");
                        break;
                  case 5:
                        System.out.println("Ви спустились на 5 поверх");
                        break;
                  case 4:
                        System.out.println("Ви спустились на 4 поверх");
                        break;
```

```
                case 3:
                    System.out.println("Ви спустились на 3 поверх");
                    break;
                case 2:
                case 1:
                    System.out.println("Ви спустились на 1 поверх");
                    break;
                default:
                    System.out.println("Ви вказали неіснуючий поверх!");
                }
            }
        }
}
```

3. Індивідуальні практичні завдання

Результат виконанням індивідуального практичного завдання необхідно завантажити в віртуальне навчальне середовище. Індивідуальне практичне завдання полягає у створенні простих програмних додатків із використанням операторів вибору if та switch.

Завдання 1. Створити програму для здійснення перевірки, чи потрапило задане ціле число (у вигляді змінної n) до проміжку (50; 100) із виводом наступного результату у консоль: «Число 120 не міститься в проміжку (25; 100)», або «Число 60 міститься в проміжку (25; 100)».

Завдання 2. Створити програму яка виводитиме у консоль найбільшу цифру заданого (у вигляді змінної n) цілого тризначного числа.

Завдання 3 (обов'язково оператор if). Створити програму, яка відтворюватиме принцип роботи пульта керування ліфтом 9-ти поверхового будинку та здійснюватиме підйом або спуск із виводом результату у консоль «Ви піднялись на 6 поверх», або «Ви спустились на 1 поверх». Для ускладнення завдання))) кнопка 2-го поверху не працює! При її натисканні під час підйому ліфт здійснюватиме доставку на 3-й поверх, і навпаки, під час спуску – на 1-й.

Це аналогічне завдання, яке виконано на практичному занятті з допомогою оператора вибору Switch. Ваше завдання реалізувати аналогічний задум тільки з використанням оператора if.

Завдання 4 (обов'язково оператор Switch). З використання оператора вибору Switch створити умовне контекстне з двома опціями: «Погодитись» та «Відмовитись». Опція «Погодитись» має спрацьовувати із подальшим виведенням у консоль стрічки «Я погоджуюсь!» при таких значеннях case: Так, ОК, Yes, Y, +, Ok. Опція «Відмовитись» має спрацьовувати із подальшим виведенням у консоль стрічки «Я відмовляють!» при таких значеннях case: Hi, NO, N, -, No.

Завдання 5* (не обов'язкове, проте дуже корисне). Оголосити та ініціалізувати три не рівних між собою цілочисельні змінні. Створити програму для сортування заданих чисел в порядку зростання.

Завдання 6* (не обов'язкове, проте дуже цікаве). На підприємстві інженер з первинними навиками програмування створив та запрограмував пристрій на екран якого виводиться кількість секунд, які залишились до кінця робочого дня. На початку робочого дня (в 9.00 ранку) на екрані висвітлює «28800 секунд» (8 годин), в обідню пору (14.30) на екрані висвітлює «9000 секунд», а по завершенню робочого дня (17.00) – на екрані «0 секунд».

Проте співробітникам підприємства було незручно оцінювати залишок робочого часу в секундах. Інженер не маючи глибоких знань в програмуванні не зміг створити програму, яка замість секунд буде виводити на екран значення **повних** годин, що залишились до кінця робочого дня, наприклад «До кінця робочого дня залишилось 2 години».

Потрібно допомогти програмісту-початківцю та реалізувати таку програму із виводом результатів у консоль. Для цього в змінну *n* потрібно записати ціле число в проміжку (0; 28800). А в результаті виконання програми на консолі має з'явитись значення у секундах та зрозуміла для співробітників стрічка про кількість **повних** годин, які містяться в секундах *n*.

Наприклад: 23466 секунд

До кінця робочого дня залишилось 6 годин

або

1249 секунд
До кінця робочого дня залишилось менше 1 годин

Файли з кодами написаних програм (розширення .java) потрібно завантажувати у віртуальне середовище (категорія контрольна частина). За бажанням, кожне завдання може завантажуватись у вигляді окремого файлу, або усі завдання в одному. Щодо місця розташування цього файлу на Вашому комп'ютерні, то це питання розглядалось в лекції «Інтегроване середовище розробки».

Практичне заняття № 5
«Написання циклічних програм»

Мета: здобути навики щодо застосування операторів циклу for, while та do-while, а також операторів переходу в процесі створення найпростіших циклічних програм.

Після практичного заняття студенти (курсанти) повинні:

вміти: створювати елементарні програми із виконанням циклічних операцій використовуючи оператори for, while та do-while елементарні програми із лінійним сценарієм виконання

знати: здобути навики щодо застосування операторів циклу for, while та do-while, а також операторів переходу в процесі створення найпростіших циклічних програм.

План заняття:

1. Написання циклічних програм використовуючи оператори for, while та do-while

2. Індивідуальне практичне завдання

1. Написання циклічних програм використовуючи оператори for, while та do-while

Для реалізації перших застосунків використовуючи оператори циклів розглянемо декілька розповсюджених прикладів.

Приклад 1: Створити програму виводу на екран значень від 1000 до 1015 використовуючи оператор циклу for:

```
public class Main {

	public static void main(String[] args) {

		for(int i = 1000; i<=1015; i++){
			System.out.println(i);
		}
	}
}
```

Далі описану умову перепишемо таким чином, щоб значення від 1000 до 1015 виводились на екран в рядок із використанням циклу while:

```
public class Main {

	public static void main(String[] args) {

		int j=1000;
		while (j<=1015){
			System.out.print(j+" ");
			j++;
		}
	}
}
```

Приклад 2: Створити програму виводу у рядок всіх непарних чисел від 1 до 55 використовуючи цикл for:

```
public class Main {
	public static void main(String[] args) {
		for (int i = 1; i<=55; i++){
			System.out.print(i++ +" ");
		}
	}
}
```

Існує й інший спосіб вирішення поставленої задачі:

```
public class Main {

	public static void main(String[] args) {
		for (int i = 1; i <= 55; i = i + 2) {
			System.out.print(i + " ");
		}
	}
}
```

Приклад 3: Створити програму виводу у консоль всіх значень від 90 до 0 з кроком у 5 за умови оголошення та ініціалізації керуючої змінної перед циклом for:

```
public class Main {

	public static void main(String[] args) {
		int i = 90;
		for(; i>=0; i=i-5){
			System.out.println(i);
		}
	}
}
```

Приклад 4: Створити програму виводу у рядок двадцяти перших значень геометричної прогресії числа 2 використовуючи оператор циклу for:

```
public class Main {

	public static void main(String[] args) {
		int n = 1;
		int m = 2;
		for(;n<=20;n++,m=m*2){
			System.out.print(m+" ");
		}
	}
}
```

Переписавши задану умову із використанням циклу while буде отримано:

```
public class Main {

	public static void main(String[] args) {

		int i = 1;
		int j = 1;
		while(i<=20){
			i++;
			j=j*2;
			System.out.print(j+" ");
		}
	}
}
```

Приклад 5: Створити циклічну програму, яка визначатиме факторіал числа 3:

```
public class Main {

	public static void main(String[] args) {

		int n,f;
		f = 1;
		n = 3;
			for(int i=1; i<=n; i++) {
				f = f*i;
			}
			System.out.println(n+"! = "+f);
	}
}
```

Приклад 6: Створити циклічну програму виводу у консоль всіх значень виразу a=2a+200 якщо виконується умова -100<i<9 або 9<i<100 (усі значення результату виконання програми – двозначні). Початкове значення керуючої змінної i = -66:

```
public class Main {

	public static void main(String[] args) {
		for(int a=-66; a<100; a=2*a+200){
			if(a>-100&&a<-9||a>9&&a<100){
				System.out.println(a);
			}
		}
	}
}
```

Приклад 7: Створити програму яка підраховуватиме кількість числових значень у діапазоні від 0 до 50000, що містять в собі «2»:

```
public class Main {

	public static void main(String[] args) {
		int f,j,k;
		j = 0;
		for(int i=1; i<=50000; i++) {
			k = i;
			f = 0;
			while(k!=0) {
				if(k%10==2) {
					f = 1;
					break;
				}
				k = k/10;
			}
			if(f == 0) {
				j++;
			}
		}
		System.out.println("Кількість числових комбінацій: "+j);
	}
}
```

2. Індивідуальне практичне завдання

Результат виконанням індивідуального практичного завдання необхідно завантажити в віртуальне навчальне середовище. Індивідуальне практичне завдання полягає у створенні простих циклічних програм із використанням операторів for, while або do-while.

Завдання 1. Створити програму виводу у консоль усіх значень від 500 до 650 з кроком 10 використовуючи усі оператори циклів for, while та do-while.

Завдання 2. Створити програму виводу у консоль усіх значень менше 5000 послідовності 2a-1, за умови що перше значення a=2.

Завдання 3. Створити програму виводу у консоль усіх додатних дільників числа 10 (дільники – цілі числа, які ділять число 10 без залишку).

Завдання 4. Створити циклічну програму визначення факторіалу числа 10 використовуючи оператори циклів for та while.

Завдання 5. Створити циклічну програму підрахунку кількості співпадінь симетричних комбінацій цифр на електронному годиннику (наприклад 03:30).

Завдання 6* (не обов'язкове, проте дуже корисне). Створити циклічну програму підрахунку кількості чисел у діапазоні від 000001 до 999999 в яких сума перших трьох цифр рівна сумі останніх трьох цифр (наприклад 003102 або 123123).

Файли з кодами написаних програм (розширення .java) потрібно завантажувати у віртуальне середовище (категорія контрольна частина). За бажанням, кожне завдання може завантажуватись у вигляді окремого файлу, або усі завдання в одному.

Практичне заняття № 6
«Застосування потоків введення/виведення та рядків в програмуванні»

Мета: здобути навики щодо роботи з основними типами даних та операторами в мові програмування Java..

Після практичного заняття студенти (курсанти) повинні:

вміти: створювати елементарні програми застосовуючи потоки введення та рядки.

знати: принцип роботи та особливості застосування потоків введення/виведення, а також рядків

План заняття:

1. Написання найпростіших програм використовуючи класи та екземпляри класів Scanner, String

2. Індивідуальне практичне завдання

1. Написання найпростіших програм використовуючи класи та екземпляри класів Scanner, String

Для реалізації перших застосунків використовуючи потоки введення/виведення та рядки розглянемо декілька розповсюджених прикладів.

Приклад 1: Створити програму виводу на екран повідомлення про парність (непарність) цілого числа введеного з клавіатури. Передбачити можливість повідомлення користувача про введення не цілого числа:

```
import java.util.Scanner;

public class Main{

    public static void main(String[] args) {

        int n;
```

```
            System.out.print("Введіть ціле число: ");
            Scanner sc = new Scanner(System.in);
            if(sc.hasNextInt()) {
                  n = sc.nextInt();
                  if(n%2==0) {
                        System.out.println("Введене число - парне");
                  } else {
                        System.out.println("Введене число - не парне");
                  }
            } else {
                  System.out.println("Ви ввели не ціле число");
            }
      }
}
```

Приклад 2: Створити програму розрахунку та виведення на екран суми двох цілих чисел введених з клавіатури. Передбачити можливість повідомлення користувача про некоректне введення чисел:

```
import java.util.Scanner;

public class Main2 {

      public static void main(String[] args) {

            int a, b, c;
            System.out.println("Для виконання програми введіть два цілих
числа");
            Scanner sc = new Scanner(System.in);
            System.out.print("Введіть перше число: ");
            if (sc.hasNextInt()) {
                  a = sc.nextInt();
                  System.out.print("Введіть друге число: ");
```

```
                    if (sc.hasNextInt()) {
                        b = sc.nextInt();
                        c = a + b;
                        System.out.print("Сума введених чисел = " + c);
                    } else {
                        System.out.print("Ви ввели не ціле число");
                    }
                } else {
                    System.out.print("Ви ввели не ціле число");
                }
            }
        }
```

Приклад 3: Створити програму визначення та виводу на екран меншого по модулю з трьох введених цілих чисел. Передбачити можливість повідомлення користувача про некоректне введення чисел:

```
import java.util.Scanner;

public class Main3 {

    public static void main(String[] args) {

        int a, b, c, moda, modb, modc, min;
        Scanner sc = new Scanner(System.in);
        System.out.print("Введіть перше ціле число: ");
        if (sc.hasNextInt()) {
            a = sc.nextInt();
            System.out.print("Введіть друге ціле число: ");
            if (sc.hasNextInt()) {
                b = sc.nextInt();
                System.out.print("Введіть третє ціле число: ");
```

```
                        if (sc.hasNextInt()) {
                              c = sc.nextInt();
                              moda = Math.abs(a);
                              modb = Math.abs(b);
                              modc = Math.abs(c);
                              if (moda <= modb && moda <= modc) {
                                    min = a;
                              } else if (modb <= moda && modb <= modc) {
                                    min = b;
                              } else {
                                    min = c;
                              }
                              System.out.println(min);
                        } else {
                              System.out.println("Ви ввели не ціле число");
                        }
                  } else {
                        System.out.println("Ви ввели не ціле число");
                  }
            } else {
                  System.out.println("Ви ввели не ціле число");
            }
      }
}
```

Приклад 4: Створити програму перевірки належності введеного слова з п’яти букв до паліндрому («ротор», «комок» тощо). Передбачити можливість повідомлення користувача про некоректне введення слова (більше або менше п’яти букв):

```
import java.util.Scanner;

public class Main4 {
```

```
	public static void main(String[] args) {


		String s1,s2;
		char a0,a1,a2,a3,a4;
		Scanner sc = new Scanner(System.in);
		System.out.print("Введіть слово з п'яти букв: ");
		if(sc.hasNext()) {
			s1 = sc.next();
			if(s1.length()==5) {
				a0 = s1.charAt(0);
				a1 = s1.charAt(1);
				a2 = s1.charAt(2);
				a3 = s1.charAt(3);
				a4 = s1.charAt(4);
				s2 = a4+""+a3+""+a2+""+a1+""+a0;
				System.out.println(s2);
				if(s1.equals(s2)) {
					System.out.println("Введене слово "+s1+" є
поліндромом");
				} else {
					System.out.println("Введене слово "+s1+" не є
поліндромом");
				}
			} else {
				System.out.print("Ви ввели слово не з п'яти букв
"+s1);
			}
		} else {
			System.out.print("Ви нічого не ввели");
		}
	}
}
```

2. Індивідуальне практичне завдання

Результат виконанням індивідуального практичного завдання необхідно завантажити в віртуальне навчальне середовище. Для виконання завдання слід нагадати теоретичний матеріал лекції. Індивідуальне практичне завдання полягає у створенні програм із застосуванням потоків введення/виведення інформації та методів класу String.

Завдання 1. З використанням методу concat() класу String скласти речення з п’яти окремих слів введених користувачем.

Завдання 2. Створити програму визначення та виведення на екран першої літери п’яти довільно введених користувачем слів із використанням методу substring(pos1, pos2).

Завдання 3. Створити програму виводу на екран повідомлення про більше дробове число з трьох введених користувачем. Передбачити можливість повідомлення користувача про введення не дробового числа.

Завдання 4. Створити програму перевірки ідентичності двох введених з клавіатури імен без урахування регістру.

Завдання 5. Створити програму визначення більшого за кількістю символів рядка з двох введених користувачем.

Файли з кодами написаних програм (розширення .java) потрібно завантажувати у віртуальне середовище (категорія контрольна частина). За бажанням, кожне завдання може завантажуватись у вигляді окремого файлу, або усі завдання в одному.

Практичне заняття № 7
«Підсумкове заняття з основ процедурного програмування»

Мета: закріпити практичні навики з основ процедурного програмування шляхом створення елементарних ігор «Вгадай число» та «Вгадай ім'я».

Після практичного заняття студенти (курсанти) повинні:

вміти: використовувати набуті навики з основ процедурного програмування.

знати: принцип роботи та особливості застосування основних операторів, умовних розгалужень, циклів, масивів, методів класу String та Math

План заняття:

1. Написання програм використовуючи основні принципи процедурного програмування.

2. Індивідуальне практичне завдання.

1. Написання програм використовуючи основні принципи процедурного програмування

Приклад 1: Створити прототип гри «Вгадай число», програма якого виконуватиме такі умови: діапазон **цілих** випадкових чисел, які генерує програма становить від 1 до 10 включно; передбачити введення числа користувачем з клавіатури; передбачити підрахунок кількості спроб до моменту відгадування числа (із виводом відповідної інформації вкінці роботи програми); за умови невірного вказування числа передбачити підказку в яку сторону (збільшення чи зменшення) слід рухатись; передбачити циклічність роботи програми до моменту відгадування числа; передбачити можливість сповіщення користувача про те, що він ввів не ціле число або ввів літеру. Код програми матиме такий вигляд:

```
import java.util.Scanner;

public class Main {
	public static void main(String[] args) {
		// prog – змінна для запису числа загаданого програмою
		// user – змінна для запису числа вказаного користувачем
		int prog, user, i;
		// Генеруємо випадкове число в проміжку від 1 до 10 включно
		do {
			prog = (int) (Math.random() * 11);
		} while (prog == 0);
```

```
            prog = (int)(Math.random()*10+1);
            // простіший варіант генерування випадкового числа не отримавши
0

            System.out.println("Програма загадала число");
            i = 0;
            user = 0;
            do {
                  i++;
                  System.out.print("Введіть число від 1 до 10 включно: ");
                  Scanner input = new Scanner(System.in);
                  if (input.hasNextInt()) {
                        user = input.nextInt();
                        if (user == prog) {
                              System.out.println("Ви вгадали!!! Кількість
спроб: " + i);
                        } else {
                              if (prog < user) {
                                    System.out.println("Ви не вгадали.
Введіть число менше");
                              } else {
                                    System.out.println("Ви не вгадали.
Введіть число більше");
                              }
                        }
                  } else {
                        System.out.println("Ви ввели не ціле число або ввели
літеру");
                  }
            } while (user != prog);
            System.out.println("До нових зустрічей!");
      }
}
```

Приклад 2: Створити прототип гри «Вгадай ім'я», програма якого виконуватиме такі умови: ім'я має складатись з 4 літер, відповідно передбачити випадкове генерування програмою ім'я саме такої величини; передбачити введення ім'я користувачем з клавіатури; передбачити можливість сповіщення користувача про введене ім'я більше або менше 4 літер; передбачити обмеження кількості спроб для вгадування ім'я до 3; передбачити сповіщення користувача про невірне вгадування ім'я із обов'язковим зазначенням кількості спроб, які залишись. Код програми матиме такий вигляд:

```
import java.util.Scanner;

public class Main1 {

	public static void main(String[] args) {

		String[] mas = { "Саша", "Маша", "Даша", "Паша", "Вітя",
"Коля", "Мотя", "Вася", "Ніна", "Ната", "Надя", "Ігор" };
		// створення масиву стрічок з переліком імен, які випадковим
		чином генеруватиме програма
		int n = (int) (Math.random() * 12);
		// генерування випадкового числа – індексу розміщення ім’я в
		масиві (від 0 до 11)
		String nameRandom = mas[n];
		String nameUser;
		// nameRandom – змінна для запису випадково згенерованого ім’я
		// nameUser – змінна для запису ім’я введеного користувачем з
		клавіатури
		int i = 0;
		int j = 3;
		// i – змінна для обліку кількості спроб
		// j – змінна для підрахунку кількості спроб, що залишились
		while (i != 3) {
			i++;

			Scanner sc = new Scanner(System.in);
			System.out.println("Комп'ютер загадав ім’я, спробуйте його
відгадати за " + j + " спроби");
			System.out.println("Введіть ім’я з 4 літер: ");
			if (sc.hasNext()) {
				nameUser = sc.next();
				if (nameUser.length() > 4) {
					System.out.println("Ви ввели ім’я більше 4
літер");
				} else if (nameUser.length() < 4) {
					System.out.println("Ви ввели ім’я менше 4
літер");
				} else {
					if (nameUser.equalsIgnoreCase(nameRandom)) {
						System.out.println("Вітаємо, ВИ
вгадали!");
						break;
					} else {

						System.out.println("Ви не вгадали! У вас
залишилось спроб:" + (3 - i));
```

```
                    }
                }
            } else {
                System.out.println("Ви ввели не ім'я");
            } j--;
        }
    }
}
```

2. Індивідуальне практичне завдання

Результат виконанням індивідуального практичного завдання необхідно завантажити в віртуальне навчальне середовище. Для виконання завдання слід нагадати теоретичний матеріал лекції. Індивідуальне практичне завдання полягає у закріплені теоретичного матеріалу останньої лекції (створення програм із застосуванням методів класу Math).

Завдання 1. Створити програму, яка виводитиме у консоль випадкове натуральне число з проміжку [-20; 20].

Завдання 2. Створити програму визначення та виводу у консоль площі та периметру прямокутного трикутника, якщо довжина катетів складає 3 та 4.

Завдання 3. Створити програму визначення та виводу у консоль значення з кількістю цифр у випадково згенерованому числі з проміжку [0; 101).

Завдання 4. Створити прототип гри «лотерея». Суть полягає у відгадуванні правильної послідовності трьох випадково згенерованих цілих чисел з проміжку [1; 3]. Обмежити користувача 2-двома спробами.

Файли з кодами написаних програм (розширення *.java) потрібно завантажувати у віртуальне середовище (категорія контрольна частина).

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Підручники та довідники

Базова

1. Основы программирования : учебник для вузов. - 4-е изд. / Иванова Г.С. – Москва : изд. МГТУ им. Н.Э. Баумана, 2007. – 416 с.

2. Изучаем Python : 4-е изд. пер. с англ. / Лутц М. – Санкт-Петербург : Символ-Плюс, 2011. – 1280 с.

3. Програмування числових методів мовою Python : підруч./ А. В. Анісімов, А. Ю. Дорошенко, С. Д. Погорілий, Я. Ю. Дорогий ; за ред. А. В. Анісімова. – К. : Видавничо-поліграфічний центр "Київський університет", 2014. – 640 с.

4. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. — М. : Издательство Юрайт, 2018. — 126 с. — (Серия : Бакалавр. Прикладной курс).

5. Легкий способ выучить Python / Зед Шоу ; [пер. с англ. М. А. Райт- мана]. — Москва : Издательство «Э», 2017. — 352 с. — (Мировой компьютерный бестселлер).

6. Основи алгоритмізації та програмування. 750 задач з рекомендаціями та прикладами : посібник / Караванова Т. П. – Київ.: «Форум», 2002 – 286 с.

7. Java 8. Полное руководство. 9-е изд.: пер. с англ. / Герберт Шилдт. – Москва : ООО "И.Д. Вильяме", 2015. – 1376 с.

8. Head First Java (изучаем Java) : пер. с англ. / Kathy Sierra, Bert Bates. – Москва : «Эксмо», 2012. – 718 с.

9. Философия Java. Библиотека ппрограмиста. 4-е изд. : пер. с англ. / Брюс Еккель : «Питер», 2009. – 640 с.

10. Дизайн-патерни – просто, як двері : підручник / Андрій Будай : «Developer’s SUCCESS», 2012. – 90 с.

11. Лямбда-выражения в Java 8. Функциональное программирование - в массы : пер. с англ. А. А. Слинкина / Ричард Уорбэртон. – Москва : ДМК Пресс, 2014. – 192 с.

12. Основи програмування на Java : методичні вказівки до лабораторного практикуму та самостійної роботи, частина перша / Бивойно П. Г., Бивойно Т. П. – Чернігів: ЧНТУ, 2014. – 79 с.

Допоміжна

1. XSLT. Справочник программиста : пер. с англ. / Кей М. – Санкт-Петербург : Символ-Плюс, 2002. –1016 с.

2. Самоучитель Python. / Д. Мусин. – СПб.: Символ – Плюс, 2017, -148 с.

3. XSLT. Справочник программиста : пер. с англ. / Кей М. – Санкт-Петербург : Символ-Плюс, 2002. –1016 с.

4. Программирование и основы алгоритмизации : Учебное пособие / В. Г. Давыдов. – Москва: Высш. шк., 2003. – 447 с.

5. Prometheus. Курс «Основи програмування». [Електронний ресурс]. – Доступний з https://edx.prometheus.org.ua/ courses /KPI

6. Prometheus. Курс «Курс CS-50». [Електронний ресурс]. – Доступний з https://edx.prometheus.org.ua/ courses /KPI

7. Програмування на Java (рос.) [Електронний ресурс]. – http://kostin.ws/java

8. Освоюємо Java. Вікіпідручник [Електронний ресурс]. – Доступний з https://uk.wikibooks.org/wiki/Освоюємо_Java

9. Java. ProgLang. Самоучитель (рос.) [Електронний ресурс]. – Доступний з http://proglang.su/java

10. Java Course (рос.) [Електронний ресурс]. – Доступний з http://javacourse.ru/begin/operations