**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра програмних систем і технологій**

**Т.В. Ковалюк**

**Методичні вказівки до лабораторних занять з дисципліни**

**«Функціональне програмування»**

для студентів спеціальності 121 «Інженерія програмного забезпечення»

освітнього рівня «бакалавр»

Київ 2020

# Загальні теоретичні відомості

### Характеристика функціональних мов програмування

Основним поняттям функціонального програмування є функція. Функціональна програма є набором визначень функцій. Функції визначаються через інші функції або рекурсивно через самих себе. У процесі виконання програми функції отримують параметри, обчислюють і повертають результат, у випадку необхідності обчислюються значення інших функцій. Програмуючи функціональною мовою, програміст не повинен описувати порядок обчислень. Йому необхідно просто описати бажаний результат у вигляді системи функцій.

Строго функціональне програмування не має операцій присвоєнь та засобів передачі керування. Повторні обчислення здійснюються за допомогою рекурсії, яка є основним засобом функціонального програмування.

Серед важливих властивостей функціонального програмування є такі:

1. Представлення програми і даних відбувається через списки. Це дає змогу програмі обробляти інші програми і навіть саму себе.
2. Функціональні мови, як правило, є інтерпретуючими мовами.
3. Функціональні мови є безтиповими, це означає, що символи не зв’язуються за замовчуванням з яким-небудь типом.
4. Функціональні мови мають незвичний синтаксис через велику кількість дужок.
5. Функціональні програми, написані для обробки символьних даних, є набагато коротші, аніж написані імперативними мовами/

### Функції-примітиви мови Scheme

Вважається, що строго функціональна мова програмування має дуже обмежене число базових функцій, на основі яких можна побудувати всі інші функції. Як правило, вибирається сім примітивних функцій. Ця обмеженість є важливою в теоретичному плані в галузі програмування, оскільки вирішується проблема розв’язання довільної задачі з допомогою обмеженого набору примітивних функцій.

Виклик функцiї у функціональних мовах програмування здійснюється у формі списку і має такий формат:

(function\_name arg1 arg2 ... argN),

Де function\_name – iм'я функцiї,

arg1,arg2,...argN – її аргументи.

Базові функції функціональної мови програмування Scheme наведено в табл. 1.

Таблиця 1. Базові функції функціональних мов програмування

|  |  |
| --- | --- |
| **Мова Scheme** | **Семантика** |
| QUOTE | Функція використовується для блокування обчислень, що призводить до інтерпретації частини S-виразу не як програми, а як даних |
| **(CAR object)** | Вибирає голову непорожнього списку, пари або неправильно-сформованого списку |
| **(CDR object)** | Вибирає хвiст непорожнього списку, пари або неправильно-сформованого спику |
| **(CONS object1 object2)** | Функція-конструктор - об'єднює об'єкти 1 та 2 |
| **(EQ? atom1 atom2)** | Порiвнює два об’єкта на тотожність |
| **LIST? object)** | Перевіряє чи є object списком. |
| (**PAIR? object)** | Перевіряє чи є object парою |
| **(СOND (condition1 action1) (condition2 action2)… (conditionN actionN))** | Набуває значення з множини значень action1, action2, ..., actionN в залежності від значень умовних виразів condition1, condition2, …, conditionN. |

Таблиця 2. Деякі математичні функції

|  |  |
| --- | --- |
| **Мова Scheme** | **Семантика** |
| **ABS** | функція знаходить модуль числа. |
| **EXP** | функція, яка повертає ex. |
| **LN** | функція, яка повертає натуральний логарифм аргумента. |
| **SIN** | функція, яка повертає синус аргумента. |
| **COS** | функція, яка повертає косинус аргумента. |
| **SQRT** | функція, яка повертає квадратний корінь аргумента. |

Таблиця 3. Предикати

|  |  |
| --- | --- |
| **NUMBER?** | предикат, який перевіряє чи аргумент є числом |
| **POSITIVE?** | предикат, який перевіряє чи аргумент є додатнім числом. |
| **NEGATIVE?** | предикат, який перевіряє чи аргумент є від’ємним числом |
| **EVEN?** | предикат, який перевіряє чи число є парним. |
| **ODD?** | предикат, який перевіряє, чи число є непарним. |
| **NULL?** | предикат, який перевіряє чи аргумент є порожнім списком. |

### Середовище функціонального програмування Dr.Racket

Оскільки в середовищі Dr.Racket підтримується ряд діалектів та стандартів, оберемо класичний стандарт Scheme R5RS. Для цього потрібно обрати меню Мова-> Обрати мову ->Other languages-> R5RS.

На рис. 1 наведено вигляд головного вікна середовища.

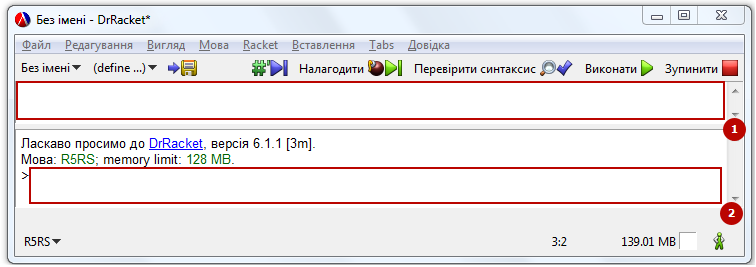


Рис. 1. Головне вікно середовища.

Головне вікно середовища розробки поділене на дві частини: 1 – область для вводу тексту програми; 2 – область командного рядка. В першій частині зручно писати текст програми і всі виклики, в другій – виклики функцій.

На панелі інструментів винесено кнопки для від лагодження, перевірки синтаксису, запуску на виконання та зупинки програми (рис. 2).



Рис. 2. Панель інструментів в середовищі Dr.Racket

Приклад 1. Обчислимо суму 1 + 2. На мові Scheme потрібно написати (+ 1 2). Після цього запустимо програму на виконання клавішою F5 або кнопкою «Запустити». Результат виконання програми відобразиться в другій половині екрану (рис. 3).

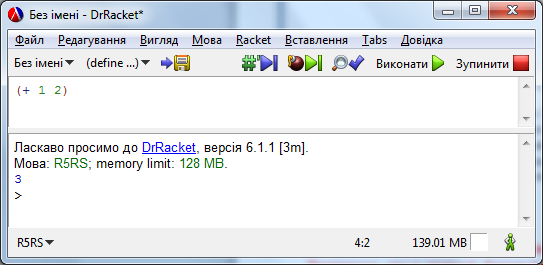


Рис. 3. Виконання програми в середовищі Dr. Racket

Є можливість виконувати команди безпосередньо в командному рядку: після введеного тексту програми потрібно натиснути клавішу Enter. Перемножимо числа 2, 3, 4: для цього введемо в командний рядок текст (\* 2 3 4) і натиснемо клавішу Enter. Наступний рядок міститиме шуканий результат (рис. 4).

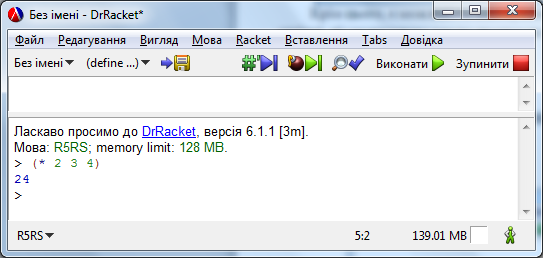


Рис. 4. Виконання програми в командному рядку

Щоб поставити коментар потрібно перед текстом поставити символ «;». Scheme не є чутливим до регістру і до кількості пробілів між об’єктами. Вирази (a) та ( a ) є тотожними.

На рис. 5 наведено приклад використання змінних та коментарів.

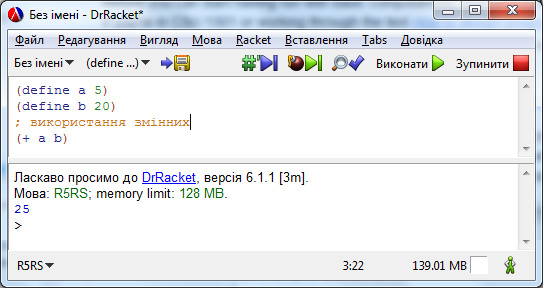


Рис. 5. Використання змінних та коментарів у середовищі Dr. Racket

### Процес налаштування коду в середовищі Dr.Racket

Приклад 2. Обчислимо вираз: **(+ 2 (\* 3 4))**. Замість кнопки «Виконати» натиснемо кнопку «Налагодити». Після цього головне вікно буде мати вигляд як на рис. 6.

Вікно має такі основні частини: 1 – текст програми, 2 – поточний результат виконання програми, 3 – меню для налаштування, 4 – вікно із відображенням вмісту стеку, 5 – вікно для відображення поточних значень змінних.

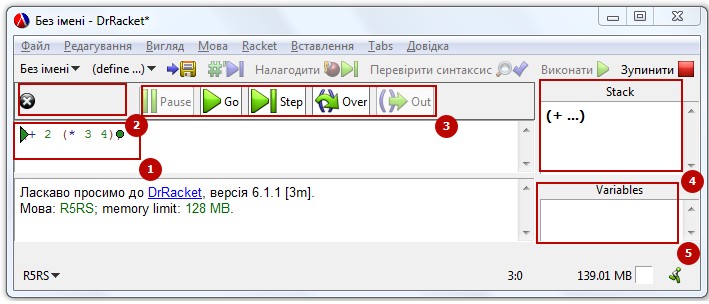


Рис. 6. Інтерфейс середовища Dr.Racket у режимі відлагодження

Спочатку в стеку є наявна функція додавання (рис. 14). Курсор відлагодження вказує на цю функцію. Натиснемо кнопку «Step». В стеку з’явилась функція множення (рис. 7). Курсор налаштування вказує на цю функцію у вікні тексту програми.

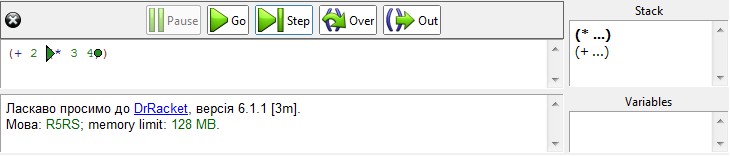


Рис. 7. Додавання в стек функції множення

Натиснемо ще раз кнопку «Step». Оскільки далі за функцією множення немає інших функцій, відбудеться обчислення множення за вказаними параметрами 3 і 4. Проміжний результат (число 12) відображено у відповідній частині екрану (рис. 16).

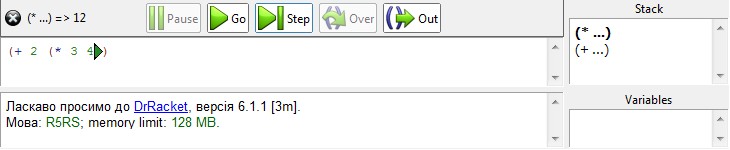


Рис. 8. Здійснення обрахунків для функції множення

Натиснемо ще раз кнопку «Step». Як видно із стеку, функція множення виконалась і вийшла з стеку. Курсор налаштування вказує на кінець виклику функції додавання. Результат виконання функцій (число 14) відображено в області біля меню налаштування (рис. 9).

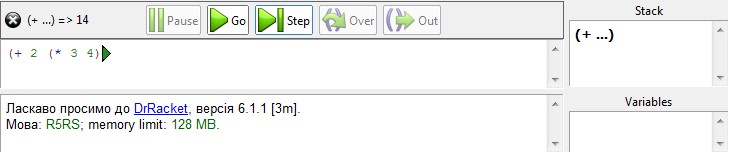


Рис. 9. Здійснення обрахунків для функції додавання

Чергове натискання кнопки «Step» завершує виконання програми – з стеку вивільняється функція додавання, а остаточний результат відображається у вікні командного рядка (рис. 18).



Рис. 18. Завершення виконання відлагодження програми

# Лабораторна робота 1. Використання рекурсії для організації повторювальних процесі

Deadline жовтень

## Теоретичні відомості

### Визначення процедур

Для створення функції з іменем потрібно використати один із форматів опису функції:

(define <name> (lambda (<arguments>) (<body>))),

або

(define (<name> <arguments>) (<body>)),

де **<name>** - ім’я функції; **<arguments>** - список аргументів (через пробіл); **<body>** - правильний S-вираз, який набуває значення.

*Приклад*.

(define (f x)

(+ x 42))

Виклик функції: (f 23)

Результат 65

Процедура, в попередньому абзаці, є абстракцією виразу за допомогою об'єктів. У першому визначенні прикладу визначена процедура, названа f. (Зверніть увагу на круглі дужки навколо f x, що позначають, що це -визначення процедури.) Вираз (f 23) є викликом процедури, приблизно означає "вирахувати (+ х 42) (тіло процедури) з x, прив'язаним до 23 ".

Оскільки процедури є об'єктами, їх можна передавати в інші процедури:

(define (f x)

(+ x 42))

(define (g p x)

(p x))

Виклик : (g f 23)

Результат: 65

У цьому прикладі тіло g обчислюється з p, прив'язаним до f, і x, що прив'язаний до 23, що еквівалентно (f 23) і обчислюється в 65.

Фактично багато операцій Scheme забезпечуються не синтаксисом, а змінними, значеннями яких є процедури. Операція +, наприклад, в Scheme є всього лише регулярним ідентифікатором, пов'язаним з процедурою, що додає числові об'єкти. Те саме стосується і \*, і багатьох інших:

(define (h op x y)

(op x y))

(h + 23 42) =) 65

(h \* 23 42) =) 966

### Хвостова рекурсія

Це випадок рекурсії, коли рекурсивний виклик функції відбувається наприкінці її роботи. Використовується у мовах програмування для оптимізації, через можливість заміни виклику функції на ітерацію, без використання стеку. Ця оптимізація широко використовується у функціональних мовах програмування.

Коли відбувається виклик функції комп'ютер має запам'ятати адресу повернення, щоб після завершення викликаної функції повернутися і продовжити виконання програми. Зазвичай адреса виконання зберігається у стеку. Іноді, остання дія функції після завершення всіх інших операцій, це просто виклик функції, можливо самої себе, і повернення результату. В цьому випадку немає необхідності запам'ятовувати адресу повернення, нововикликана функція буде повертати результат безпосередньо за адресою повернення записаною для початкової функції.

Приклад на [Scheme](https://uk.wikipedia.org/wiki/Scheme):

(**define** (factorial n)

(**define** (fac-times n acc)

(**if** (= n 0)

acc

(fac-times (- n 1) (\* acc n))))

(**if** (< n 0)

(display "Невірний параметр!")

(fac-times n 1)))

\

Звичайний рекурсивний спосіб обчислення факторіала, наведений нижче, **не** є хвостовим-рекурсивним, оскільки в кожному виклику функції після рекурсивного виклику виробляються додаткові операції, а саме множення на *n.*

Приклад на [Scheme](https://uk.wikipedia.org/wiki/Scheme):

( define ( factorial n )

( if ( = n 0 )

1

( \* n ( factorial ( - n 1 ) )

) ) )

## Завдання до лабораторної роботи 1

**Написати процедури, що обчислюють задану функцію за допомогою рекурсивного процесу. Продемонструвати застосування звичайної та хвостової рекурсії.**

|  |  |
| --- | --- |
| **Номер варіанта** | **Умова завдань** |
|  | 1.1 Ввести з клавіатури натуральне число *n*. Знайти суму його цифр, використовуючи рекурентне означення функції *f(n)*:    *Умова продовження рекурсії:* сума цифр числа дорівнює останній цифрі плюс сума цифр числа без останньої цифри (числа, що ділиться без остачі на 10).  *Умова закінчення рекурсії:* якщо число дорівнює 0, то сума його цифр дорівнює 0. |
| * 1. Вкладник поклав в банк *sum* грошових одиниць під *pr* відсотків за один період. Усі дані вводити з клавіатури. Використовуючи рекурсію, визначити величину вкладу по звершенні *m* періодів часу.   **Контрольний тест**: введені дані: сума вкладу 1000, відсотки за період 1.25, кількість періодів 12, отриманий результат: 1160.75 |
|  | 2.1 Ввести з клавіатури два натуральних числа *n* та *m*. Обчислити кількість комбінацій з *n* різних елементів по *m*. Кількість комбінацій визначається рекурентним співвідношенням:  де  біноміальні коефіцієнти, які розраховують за формулою Визначити глибину рекурсії. |
| 2.2 Увести з клавіатури два цілих числа *A* і *В*. Використовуючи рекурсію, вивести всі числа від *A* до *В* включно в порядку спадання, якщо *A* більше *В* або в порядку зростання в іншому випадку. **Контрольний тест:** введені числа 7 3, отриманий результат: 7 6 5 4 3. |
|  | 3.1. Ввести з клавіатури два натуральних числа *n* та *m*. Розрахувати значення функції Аккермана *A(m, n)* та глибини рекурсії, використовуючи рекурентне співвідношення:    Визначити глибину рекурсії. **Контрольний тест**: *А*(2, 2) = 7. |
| 3.2 Увести з клавіатури натуральне число *n* >1. Вивести всі прості дільники цього числа в порядку неспадання з урахуванням кратності. Алгоритм повинен мати складність O (√*n*). **Контрольний тест:** введено число 18, отриманий результат: 2 3 3. |
|  | 4.1 Ввести з клавіатури два цілих додатних числа *X* та *N*. Звести число *X* в степінь *N*, використавши рекурентне співвідношення:    Визначити глибину рекурсії. |
| 4.2 Увести з клавіатури натуральне число *n*. Обчисліть суму його цифр. **Контрольний тест:** введено 179, отриманий результат: 17. |
|  | 5.1 Ввести з клавіатури два натуральних числа *m* та *n*. Знайти їх найбільший спільний дільник, застосувавши алгоритм Евкліда (GCD - Greatest Common Divisor) для рекурентного співідношення:    Запис *a* mod *b* означає остачу від ділення *a* на *b*. *K* - натуральне число, що означує кратність чисел *a* та *b.* Визначивши глибину рекурсії. |
| 5.2 Увести з клавіатури натуральне число *n*. Вивести всі його цифри по одній в зворотному порядку, розділяючи їх пробілами або новими рядками. При розв'язанні цього завдання дозволена тільки рекурсія і целочисельна арифметика. **Контрольний тест**: введено число 123, отриманий результат: 3 2 1. |
|  | 6.1. З *n* солдатів, вишикуваних в шеренгу, потрібно відібрати кількох в розвідку. Для здійснення цього виконується наступна операція: якщо солдат в шерензі більше ніж 3, то видаляються всі солдати, які стоять на парних позиціях, або всі солдати, які стоять на непарних позиціях. Ця процедура повторюється до тих пір, поки в шерензі залишиться 3 або менше солдатів. Їх і відсилають в розвідку. Обчислити кількість способів, якими можуть бути сформовані групи розвідників рівно з трьох осіб. Кількість солдатів *n* вводиться з клавіатури. Рекурентне співвідношення для обчислення кількості способів *f(n)*, якими можна сформувати групи розвідників з *n* осіб в шерензі, таке:    Визначити глибину рекурсії. |
| 6.2 Увести з клавіатури натуральне число *n*. Вивести всі його цифри по одній в прямому порядку, розділяючи їх пробілами або новими рядками. При розв'язанні цього дозволена тільки рекурсія і целочисельна арифметика. **Контрольний тест:** введено число 123, отриманий результат: 1 2 3. |
|  | 7.1. Ввести з клавіатури два цілих додатних числа m>0 та n>0. Розробити рекурсивну функцію, яка обчислює площу трикутника на основі рекурентного співвідношення:    Визначити глибину рекурсії. |
| 7.2 Увести з клавіатури натуральне число *n* > 1. Перевірити, чи є воно простим. Програма повинна вивести слово YES, якщо число просте і NO, якщо число складене. Алгоритм повинен мати складність O (√*n*). Для застосування рекурсії для цієї задачі потрібно робити рекурсію за параметром, яким є дільник числа. **Контрольний тест**: введено число 6, отриманий результат: NO. |
|  | 8.1 Ввести з клавіатури два цілих додатних числа a та b. Обчислити добуток двох цілих додатних чисел за рекурентним співвідношенням:    Визначити глибину рекурсії. |
| 8.2.Увести з клавіатури два натуральних числа *k* та *s*. Визначити кількість k-значних натуральних чисел, сума цифр яких дорівнює *s*. Запис натурального числа не може починатися з цифри 0. В цьому завданні можна використовувати цикл для перебору всіх цифр, що стоять на будь-якій позиції. **Контрольний тест:** введені числа 3 15, отриманий результат: 69. |
|  | 9.1. По колу стоять *n* людей, яким присвоєні номери від 1 до *n*. Починаючи відлік з першого і рухаючись по колу, кожна друга людина виходитиме з кола доти, поки не залишиться одна. Нехай номер того, хто залишився, *x*. Потім по колу стоятиме *x* людей і процедура виходу з колу людей повторюватиметься доти, поки не залишиться одна людина з номером *y*. Ці процедури повторюватимуться доти, поки номер тої людини, що залишиться, не стане рівним первинній кількості людей в потоковому раунді. Визначити номер людини, яка залишилася, і кількість повторів процедури. Номер людини *f(n)*, що залишилася, обчислюється за рекурентним співвідношенням:    Визначити глибину рекурсії. |
| 9.2. Увести з клавіатури натуральне число *n*. Вивести слово YES, якщо число *n* є точним степенем двійки, або слово NO в іншому випадку. **Операцією зведення в степінь користуватися не можна!**. **Контрольний тест:** ввведено число 8, отриманий результат: YES. |
|  | 10.1. Біноміальні коефіцієнти розкладання бінома утворюють *і*-й рядок трикутника Паскаля. Кожне число в трикутнику, крім перших трьох, є сумою чисел, розташованих над ним у попередньому рядку. Число в *і*-му рядку (*і* = 0, 1, 2, …) на *j*-му місці (*j* = 0, 1, …, i) задається формулою . Сам трикурник Паскаля задається рекурентним співвідношенням.    «Верхівка» трикутника має наступний вигляд. Надрукувати перші *n* рядків трикутника Паскаля, використавши рекурсивне визначення його елементів. Кількість рядків вводиться з клавіатури. Визначити глибину рекурсії. |
| 10.2 Увести з клавіатури два натуральних *a* і *b*. Визначити кількість послідовностей з *a* нулів і *b* одиниць, в яких ніякі два нулі не стоять поруч. Вивести знайдені послідовності з нулів та одиниць. **Контрольний тест**: введені числа 2 2, отриманий результат: 3 0101 1010 0110. |
|  | 11.1.Послідовність 1, 1, 2, 3, 5, 8, ... складається з чисел Фібоначчі. Кожен елемент, починаючи з третього, дорівнює сумі двох попередніх. Рекурентне співвідношення для розрахунку чисел Фібоначчі таке:    Ввести з клавіатури два натуральних числа *m* та *n*, які означають кількість чисел та номер числа в послідовності Фібоначчі. Вивести послідовність чисел Фібоначчі в кількості *m* елементів та значення *n*-го числа. Передбачити випадок *m* < *n*. Визначити глибину рекурсії. |
| 11.2 Створити рекурсивну функцію, яка отримує числа, зчитуючи їх з клавіатури, і перевіряє їх на непарність. Кінець вводу - число 0. Функція не повертає значення, а відразу ж виводить результат на екран, зберігаючи порядок ведених чисел. У цьому завданні не можна використовувати глобальні змінні і передавати будь-які аргументи в рекурсивну функцію. Основна програма повинна складатися тільки з виклику цієї функції. **Контрольний тест:** введені числа 3 2 1 0, отриманий результат: 3 1. |
|  | Ввести з клавіатури два натуральних числа *n* та *k*. Кількість розміщень без повторень  і кількість сполучень без повторень  можуть бути знайдені відповідно за формулами , . Застосувати формули для розрахунку кількості розміщень і сполучень для визначення кількості способів розподілу *k* обов'язків між *n* членами комісії та кількості способів розподілу уроків в *n* класах між трьома вчителями, якщо кожен учитель викладатиме у *k* класах. Визначити глибину рекурсії. |
| 12.2. Увести з клавіатури натуральне число *n*, десятковий запис якого не містить нулів. Отримайте число, записане тими самими цифрами, але в протилежному порядку. При розв'язанні цього завдання **не можна** використовувати цикли, рядки, списки, масиви, дозволяється тільки рекурсія і целочисельна арифметика. Функція повинна повертати ціле число, яке є результатом роботи програми, виводити число по одній цифрі **не можна**. **Контрольний тест**: введено число 179, отримали результат: 971. |
|  | 13.1 Ввести з клавіатури два натуральних числа для обчислення кореня *k*>1 -ого степеня з числа *a*>=0, використовуючи рекурентну формулу  , де *n* - кількість елементів послідовності наближених значень кореня *k*-ого степеня з числа *a*. Визначивши глибину рекурсії. |
| 13.2 Увести з клавіатури два натуральних числа *m* в десятковій системі числення та *n* в двійковій системі. Використовуючи рекурсивну функцію, перевести число *m* з десяткової системи числення в двійкову, а число *n* з двійкової системи числення в десяткову. **Контрольний тест**: введені числа 25 10101, отриманий результат 11001 21. |
|  | 14.1. Увести з клавіатури три натуральних числа *b, p, m*. Обчислити значення виразу , де операція *mod* обраховує остачу від ділення цілих чисел. Для зведення в степень *b^p* з логарифмічною складностю *O(log p)* використати рекурентне співвідношення:    Визначити глибину рекурсії. |
| 14.2. Увести з клавіатури натуральне число *n*. Використовуючи рекурсивну функцію, визначити і вивести всі непарні (парні) числа з послідовності цілих чисел від *n* до 0, зберігаючи їх порядок. Контрольний тест: введено число 8, отриманий результат: 7 5 3 1. |
|  | 15.1. У речовій лотереї розігруються *m* предметів. Усього в урні *n* квитків. Виймається *k* квитків. Значення *m, n, k* вводять з клавіатури. Скількома способами квитки можна вийняти з урни так, щоб: а) рівно два з них були виграшними, б) принаймні два з них були виграшними? Кількість способів вибору квитків визначається формулою сполучень , комбінаторними правилами суми та добутку. Реалізувати рекурсивний варіант розв'язку задач. Визначивши глибину рекурсії. |
| 15.2. Монотонною послідовністю називається послідовність натуральних чисел, в якій кожне натуральне число *k* зустрічається рівно k раз: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4 ... Ввести з клавіатури натуральне число *n*. Використовуючи рекурсію, вивести перші *n* членів цієї послідовності. Контрольний тест: введено число 15, отриманий результат: 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5. |
|  | 16.1. Визначити кількість розбиття *p(n,m)* додатного цілого числа *n*, значення якого ввести з клавіатури. Розбиття цілого числа – це його зображення у вигляді суми цілих додатних чисел. Кількість розбиття цілого числа *n* із доданками, що не перевищують значення *m*, визначається за рекурентним співвідношенням:    Реалізувати рекурсивний варіант розв'язку задачі, Визначивши глибину рекурсії. |
| 16.2. Увести з клавіатури два натуральних *m* і *n*. Використовуючи рекурсію, визначити найменше спільне кратне (НСК) введених чисел. НСК двох цілих чисел *m* і *n* є найменше натуральне число, яке ділиться на *m* і *n* без залишку, тобто кратне їм обом. НСК(*m*, *n*)=(*m*\* *n*)/НСД(*m*, *n*), де НСД - найменший спільний дільник. **Контрольний тест:** введені числа 16 20, отриманий результат: 80. |
|  | 17.1. Увести з клавіатури два цілих числа *p, q*. Обчислити суму функцій *f(n)* за введеними значеннями *p, q* за формулою . Функції *f(n)* при *n*, що змінюється від *p* до *q,* визначена через рекурентне співвідношення:    Реалізувати рекурсивний варіант розв'язку задачі. Визначити глибину рекурсії. |
| 17.2. Увести з клавіатури натуральне число *n*. Вивести всі прості множники цього числа в довільному порядку з урахуванням кратності. **Контрольний тест:** уведено число 84, отриманий результат: 2 2 3 7. |
|  | 18.1. Числа Фібоначчі визначаються рекурентним співвідношенням:    Біноміальні коефіцієнти  розраховуються за рекурентним співвідношенням:    Довести, що для перших чисел Фібоначчі справедлива формула: . Реалізувати рекурсивний варіант розв'язку задачі. Визначити глибину рекурсії. |
| 18.2. Увести з клавіатури натуральне число *n*. Вивести всі числа від 1 до *n*, використавши рекурсію. **Контрольний тест:** введено число 5, отриманий результат: 1 2 3 4 5 |
|  | 19.1. Ввести ціле число *n* в десятковій системі числення з клавіатури. Перевести його у двійкову систему. Знайти кількість одиниць в двійковому представленні числа *n*, використовуючи рекурентне означення функції *f(n)*, де символ & означує операцію побітового логічного множення:    Реалізувати рекурсивний варіант розв'язку задачі. Визначити глибину рекурсії. |
| 19.2. Потрібно сплатити поштове відправлення, вартість котрого складає *m* копійок, а в наявності тільки поштові марки номіналом *x, y, z* копійок. Скількома різними способами можна сплатити поштове відправлення? Розробити рекурсивну функцію для обчислення кількості зображень числа *m* у вигляді суми певних фіксованих чисел з використанням рекурентних співвідношень. Використати рекурентне співвідношення для чисел Фібоначчі. |
|  | 20.1.Увести з клавіатури ціле число *n* та дійсне *x*. Обчислити значення поліному від *x* степені *n* за рекурентним співвідношенням:    Реалізувати рекурсивний варіант розв'язку задачі. Визначити глибину рекурсії. |
| 20.2. Використовуючи рекурсивні функції, перевірити, чи можна задане з клавіатури натуральне число представити у вигляді: а) добутку двох простих чисел; б) добутку трьох простих чисел; в) квадрата будь-якого простого числа; г) куба будь-якого простого числа. |

# Лабораторна робота 2. Рекурентні співвідношення для тригонометричних, експоненціальних функцій та ланцюгові дроби

Deadline жовтень

## Теоретичні відомості

Формула, що виражає член послідовності через один або декілька попередніх, називається *рекурентним співвідношенням*. Послідовність, члени якої задовольняють деякому рекурентному співвідношенню, називається рекурентною.

У загальному випадку рекурентне співвідношення визначає залежність члена послідовності {*Sn*}від *k* попередніх членів: *Sn = F(Sn-k,…,Sn-1)*.

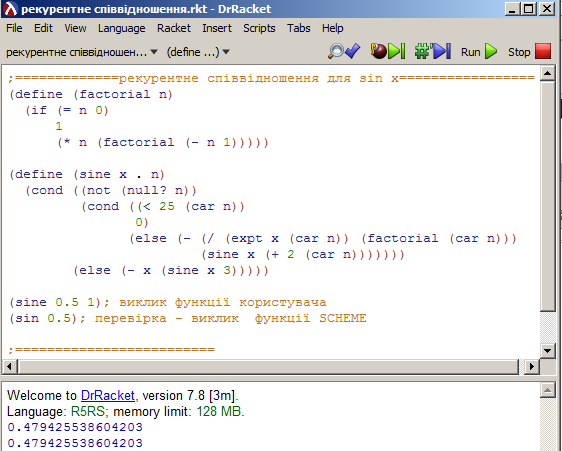
Наближене значення суми ряду можна отримати або обмежуючись сумою перших *n* його членів, або обчислюючи суму з наперед заданою точністю. Формула загального члена даного ряду є достатньо простою, але використовувати її не раціонально, оскільки для кожного члена ряду треба обчислювати степінь і факторіал. Набагато вищої ефективності можна досягти, обчислюючи член ряду за допомогою рекурентного співвідношення. Найпростішими прикладами рекурентних послідовностей є арифметична та геометрична прогресії, елементи яких пов’язані з попередніми елементами співвідношеннями *an = an*-1+d та *an = an*-1×q, де *d* та *q* — деякі сталі величини, *an* *–* значення елемента ряду на кроці *n.*

Із заданою точністю може бути обчислена сума лише збіжного ряду, а довільний степеневий ряд має певну область збіжності (можливо, порожню), тобто збігається не за всіх, а лише за деяких значень *x* (ряд, що розглядається нами як приклад, збігається для будь-якого дійсного *x*). По-друге, простий спосіб перевірки точності часткової суми ряду існує не для всіх рядів. Такий спосіб існує, зокрема, для знакозмінних рядів, абсолютні величини членів яких, починаючи з деякого номера, утворюють монотонно спадну послідовність. Для таких рядів сума всіх членів, починаючи від (*n*+1)-го, є меншою за модулем від *n*-го: .

### Рекурентні формули розвинення функцій у ряди

|  |  |
| --- | --- |
| **Функція** | **Розвинення у ряд Маклорена (Тейлора)** |
| sin x | *x* – *x*3/3! +*x*5/5! – x7/7!+…. |
| cos x | 1 – *x*2/2! + *x*4/4! – *x*6/6!+… |
| arcsin x |  |
| arctg x | *x* –*x*3/3 + *x*5/5 – *x*7/7+… |
| sh x | *x* +*x*3/3! + *x*5/5! + *x*7/7!+… |
| ch x | 1 + *x*2/2! + *x*4/4! + *x*6/6!+… |
| ln x | (*x*–1) – (*x*–1)2/2+(*x*–1)3/3– (–1)4/4+…. |
| ex | 1 + *x*/1! + *x*2/2! + *x*3/3!+… |
| e-x | 1 – *x*/1! + *x*2/2! – *x*3/3!+… |
| √x | *y0* = 1, *yn+1* = 1/2\*(*yn*+*x*/*yn*), *n* = 0,1,2,… |

Приклад коду



## Завдання до лабораторної роботи 2

1. Написати процедури, що обчислюють задану функцію за допомогою рекурентних послідовностей, розвинувши її у ряд Маклорена (абоТейлора).
2. Параметр функції має змінюватися від заданого в процесі виклику мінімального значення до максимального значення із певним кроком.
3. Розвинення функції в ряд здійснювати із заданою точністю. Точність розрахунку задавати в діапазоні від 10-2 до 10-6.
4. Для розвинення функції у ряд Маклорена (абоТейлора) створити власну функцію, яка розраховує суму ряду за рекурентним співвідношенням.
5. Значення функції tg(x) обчислювати через функції sin(x) та cos(x).
6. Визначити похибку обчислення наближених значень функції як різницю абсолютних значень наближеного обчислення та стандартного значення функції.
7. Стандартне значення функції обчислювати за допомогою бібліотечних математичних функцій.

|  |  |
| --- | --- |
| **Номер варіанта** | **Умова завдань** |
| 1 | 1.1 Обчислити значення функції у, розвинувши функцію cos(x) у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. Визначити похибку. J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var1.gif |
| 1.2. Обчислити нескінчений ланцюговий дріб, задавши значення *n* при виклику функції  F:\Kovalyuk500\!!KHУ Шевченка\!COURSES=DISCIPLINES\основи програмування С&C++\C&C++\MethodLabWorkC++\labs\lab_04\pictures\var2_1.gif |
| 2 | 2.1. Обчислити значення функції у, розвинувши функцію у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. Визначити похибку.  J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var2.gif |
| 2.2. Обчислити скінчений ланцюговий дріб, задавши значення *а* при виклику функції  F:\Kovalyuk500\!!KHУ Шевченка\!COURSES=DISCIPLINES\основи програмування С&C++\C&C++\MethodLabWorkC++\labs\lab_04\pictures\var2_2.gif |
| 3 | Обчислити значення функції у, розвинувши функцію е-х у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. Визначити похибку.J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var3.gif |
| * 1. Обчислити нескінчений ланцюговий дріб, задавши значення точностіпри виклику функції. Ланцюговий дріб виражає число π.   F:\Kovalyuk500\!!KHУ Шевченка\!COURSES=DISCIPLINES\основи програмування С&C++\C&C++\MethodLabWorkC++\labs\lab_04\pictures\var2_3.gif |
| 4 | Обчислити значення функції у, розвинувши функцію sin(x) у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. ВизначитиJ:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var4.gif |
| * 1. Обчислити нескінчений ланцюговий дріб, задавши значення точностіпри виклику функції   F:\Kovalyuk500\!!KHУ Шевченка\!COURSES=DISCIPLINES\основи програмування С&C++\C&C++\MethodLabWorkC++\labs\lab_04\pictures\var2_4.gif |
| 5 | 5.1.Обчислити значення функції у, розвинувши функцію ln(x) у ряд Тейлора. Аргумент х змінюється від -1 до 3 з кроком 0.5. Визначити похибкуJ:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var5.gif |
| * 1. Обчислити нескінчений ланцюговий дріб, задавши значення *точності* при виклику функції.   F:\Kovalyuk500\!!KHУ Шевченка\!COURSES=DISCIPLINES\основи програмування С&C++\C&C++\MethodLabWorkC++\labs\lab_04\pictures\var2_5.gif |
| 6 | 6.1. Обчислити значення функції у, розвинувши функцію ln(x) у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. Визначити похибку |
| 6.2. Обчислити нескінчений ланцюговий дріб, задавши значення *точності* при виклику функції  F:\Kovalyuk500\!!KHУ Шевченка\!COURSES=DISCIPLINES\основи програмування С&C++\C&C++\MethodLabWorkC++\labs\lab_04\pictures\var2_6.gif |
| 7 | 7.1. Обчислити значення функції у, розвинувши функцію cos(x) у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. Визначити похибку J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var11.gif |
| 7.2. Обчислити нескінчений ланцюговий дріб, задавши значення *точності* при виклику функції  F:\Kovalyuk500\!!KHУ Шевченка\!COURSES=DISCIPLINES\основи програмування С&C++\C&C++\MethodLabWorkC++\labs\lab_04\pictures\var2_7.gif |
| 8 | 8.1. Обчислити значення функції у, розвинувши функцію у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. Визначити похибку. J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var13.gif |
| 8.2. Обчислити нескінчений ланцюговий дріб, задавши значення *точності* при виклику функції  F:\Kovalyuk500\!!KHУ Шевченка\!COURSES=DISCIPLINES\основи програмування С&C++\C&C++\MethodLabWorkC++\labs\lab_04\pictures\var2_8.gif |
| 9 | 9.1. Обчислити значення функції у, розвинувши функцію tg(x) у ряд Тейлора подавши її через sin(x) та cos(x).. Аргумент х змінюється від -3 до 3 з кроком 0.5. Визначити похибкуJ:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var17.gif |
| 9.2. Обчислити для заданого натурального числа *n* вираз:  F:\Kovalyuk500\!!KHУ Шевченка\!COURSES=DISCIPLINES\основи програмування С&C++\C&C++\MethodLabWorkC++\labs\lab_04\pictures\var2_9.gif |
| 10 | 10.1. Обчислити значення функції у, розвинувши функцію ln(1+x) у ряд Тейлора. Аргумент х змінюється від -3 до 3 з кроком 0.5. Визначити похибку |
| 10.2. Обчислити для заданого натурального числа *n* вираз:  F:\Kovalyuk500\!!KHУ Шевченка\!COURSES=DISCIPLINES\основи програмування С&C++\C&C++\MethodLabWorkC++\labs\lab_04\pictures\var2_10.gif |
| 11 | 11.1Обчислити значення функції у, розвинувши функцію cos(x) у ряд Тейлора. Аргумент х змінюється від -2 до 3 з кроком 0.5. Визначити похибку J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var15.gif |
| 11.2. Ввести з клавіатури натуральне число *n* . Необхідно отримати всі досконалі числа, менші за *n* . Досконалим називається число, значення якого дорівнює сумі всіх його дільників. |
| 12 | 12.1Обчислити значення функції у, розвинувши функцію e-x у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. Визначити похибку.J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var21.gif |
| 12.2 Ввести з клавіатури натуральне число *n* . Визначити кількість цифр у числі *n* , підрахувати їх суму та знайти першу і останню цифри числа. Подання числа у вигляді структурованого типу (масивом або рядком) **не використовувати** . |
| 13 | 13.1. Обчислити значення функції у, розвинувши функцію ln(x) у ряд Тейлора. Аргумент х змінюється від -2 до 4 з кроком 0.5. Визначити похибку. J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var23.gif |
| 13.2. Ввести з клавіатури два натуральних числа *m>100, n> m* . Визначити кількість чисел між *m, n*, які складаються з непарних цифр, або мають різні цифри. Подання числа у вигляді структурованого типу (масивом або рядком) **не використовувати** . |
| 14 | 14.1. Обчислити значення функції у, розвинувши функцію у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. Визначити похибку. J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var2.gif |
| 14.2. У трьох видах спорту стартує***n***осіб. Визначити, скільки існує можливих результатів, якими можуть закінчитися змагання, якщо кожна людина стартує в одному, довільно обраному виді спорту? Під результатом змагання розуміється розподіл місць для всіх спортсменів, що стартують в кожному виді. |
| 15 | 15.1. Обчислити значення функції у, розвинувши функцію sin(x) у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. ВизначитиJ:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var4.gif |
| 15.2. Учасники лижних змагань стартують з інтервалом 30 секунд. Щоб визначити порядок старту, спортсмени тягнуть жереб, який вказує номер старту. Визначити кількість різних послідовностей виходу лижників на старт, якщо в змаганнях брало участь *n* осіб. Через який проміжок часу всі спортсмени будуть на лижах? Значення кількості спортсменів *n* вводити з клавіатури. |
| 16 | 16.1. Обчислити значення функції у, розвинувши функцію arctg(x) у ряд Тейлора. Аргумент х змінюється від 0 до 3 з кроком 0.5. Визначити похибку J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var6.gif |
| 16.2. Два натуральних числа називаються "дружніми", якщо кожне з них дорівнює сумі всіх дільників іншого, крім самого цього числа. Визначити всі пари "дружніх" чисел, що лежать у діапазоні [200, 300] . |
| 17 | 17.1. Обчислити значення функції у, розвинувши функцію ex у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. Визначити похибку. J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var8.gif |
| 17.2. Натуральне число із *n* цифр є числом Армстронга, якщо сума його цифр, піднесених до *n*-го степеня, дорівнює самому числу (наприклад, 153 = 1^3 + 5^3 + 3^3=1+125+27). Визначити всі числа Армстронга, що складаються з двох, трьох та чотирьох цифр |
| 18 | 18.1. Обчислити значення функції у, розвинувши функцію sin(x) у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. Визначити похибку J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var10.gif |
| 18.2. Для заданого натурального числа *n* визначити всі Піфагорові трійки натуральних чисел, кожне з яких не перевищує *n* . Піфагорові трійки натуральних чисел *a, b, c* відповідають умові: *a^2+b^2=c^2*, (*a<=b<=c<=n*) . |
| 19 | 19.1. Обчислити значення функції у, розвинувши функцію e-x у ряд Тейлора. Аргумент х змінюється від -2 до 2 з кроком 0.5. Визначити похибку. J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var12.gif |
| 19.2. Ввести з клавіатури два натуральних числа *m, n*. Визначити кількість способів, якими можна розсадити *n* людей серед *m* людей за круглим столом. Розміщення, що відрізняються тільки циклічним переміщенням навколо столу, вважаємо однаковими. |
| 20 | 20.1. Обчислити значення функції у, розвинувши функцію sh(x) у ряд Тейлора. Аргумент х змінюється від -3 до 3 з кроком 0.5. Визначити похибку. J:\!TeachingStudentKPI2018\course1_A&P\A&Psemestr1\LabA&P_semestr1\MethodLabWorkC++Semestr1Html\labs\lab_04\pictures\var20.gif |
| 20.2. Обчислити скінчений ланцюговий дріб, задавши значення *n* при виклику функції |

# Лабораторна робота 3. Форми lambda та let. Вираз присвоєння set!

Deadline жовтень

## Теоретичні відомості

Для визначення нової функції мовою Scheme використовується така форма:

(lambda (<arguments>) (<body> )),

де **<arguments>** - список аргументів (через пробіл); **<body>** - правильний S-вираз, який набуває значення.

Такий запис функцій називається лямбда-функцією, оскільки функція немає імені для її зручного виклику. Наприклад, визначемо функцію одного аргументу, яка додає до переданого аргументу число 10:

( lambda ( x ) ( + x 10 ) )

Згідно концепції Lisp, всі функції викликаються у форматі

(function\_name param1 param2 … paramN)

Якщо функція є без імені, то можна підставити замість **function\_name** сам опис функції. Наприклад, виклик вищеописаної функції для параметру 20:

( ( lambda ( x )( + x 10 ) ) 20 )

Недоліком такого опису функції є складність її виклику, оскільки для цього потрібно завжди використовувати повний опис функції.

Для створення функції з іменем потрібно використати один із форматів опису функції:

(define <name> (lambda (<arguments>) (<body>)))

де **<name>** - ім’я функції; **<arguments>** - список аргументів (через пробіл); **<body>** - правильний S-вираз, який набуває значення.

Приклад.

((lambda(x)  
 (+ x 42))  
 23)

Правилами для ключового слова lambda визначено, що перша підформа є списком параметрів, а решта підформи - тілом процедури.

Змінні Scheme, прив'язані визначеннями або виразами **let** або **lambda**, фактично прив'язуються не безпосереднє до об'єктів, що визначені у відповідних прив’язках, а до адреси пам'яті, що містить ці об'єкти. Вміст за цими адресами згодом може бути змінено за допомогою присвоювання:

;example1

(let ((x 23))  
 (set! x 42)

;example2

(let ((a 2)  
 (b 3)  
 (+ a b))

У даному випадку тіло виразу let складається з двох обчислюваних послідовно виразів зі значенням остаточного виразу, що приймає значення всього виразу let. Вираз (set! х 42) є присвоєнням, що вказує "замінити об'єкт за адресою, на яку вказує x, на 42 ". Таким чином, попереднє значення x, що дорівнює 23, змінюється на 42.

### Методи розв’язання нелінійних рівнянь

#### Метод ділення відрізка навпіл (метод бісекції)

Для знаходження кореня рівняння *f(x)*=0, що належить відрізку [*a*, *b*], ділимо цей відрізок навпіл. Якщо , то  є коренем рівняння. Якщо , то вибираємо ту з половин  або , на кінцях якої функція *f(x)* має протилежні знаки. Новий звужений відрізок  знову ділимо навпіл і робимо ті самі дії.

#### Метод хорд

При розв'язанні нелінійного рівняння *f(x)*=0 методом хорд задаються відрізок [*a, b*], на якому існує тільки один розв'язок, і точність ε . На 0 -й ітерації методу [*a0, b0*] [*a, b*], на *k* -й ітерації методу маємо поточний відрізок [*ak, bk*]. Потім через дві точки з координатами (*ak* , *f*(*ak*)) й [*bk*, f(*bk*)] проводимо відрізок прямої лінії (хорду) і визначаємо точку перетину цієї лінії з віссю абсцис (точка *xk*). Якщо при цьому *f*(*ak* )× *f*(*xk* ) < 0 , то праву межу інтервалу переносимо в точку *xk* (тобто *bk*+1 = *xk* , *ak* +1 = *ak* ). Якщо зазначена умова не виконується, то в точку *xk* переноситься ліва межа інтервалу (*ak* +1 = *xk* , *bk* +1 = *bk* ). Пошук розв'язку припиняється при досягненні заданої точності, тобто *f*(*xk*) ( ) < ε. Точка перетину хорди з віссю абсцис визначається за формулою:



#### Метод дотичних (метод Ньютона).

При розв'язанні нелінійного рівняння *f(x)*=0 методом дотичних задаються початкове наближення *x*0 і точність ε. Потім у точці (*x*0, *f*(*x*0)) проводиться дотична до графіка *f*(*x*) й визначається точка *x*1 перетину дотичної з віссю абсцис. У точці( *x*1,*f*(*x*1)) знову будується дотична, обчислюється наступне наближення шуканого розв'язку *x*2 і т. д. Зазначена процедура повторюється доти, поки |*f*(*xi*)|> ε . Точка перетину (*k* +1)-ої дотичної з віссю абсцис визначається за формулою:



Умова збіжності методу дотичних *f*(*x0*)× *f‘*(*x0*)>0

#### Метод простої ітерації

Метод простої ітерації полягає в тому, що рівняння *f(x)*=0 попередньо приводиться до канонічного вигляду *x* = φ(*x*). Ітерації виконуються за правилом xk+1=φ(xk) для *k*=0,1,…. Зазначена процедура припиняється при досягненні заданої точності, тобто |*f*(*xi*)| = |*xk* – φ(*xk*)|≤ ε. Умова збіжності методу ітерацій:|φ′(*x*)| ≤ q < 1 для всіх *x* ∈[*a*,*b*] .

### Чисельне інтегрування функцій одної змінної

#### Формула Сімпсона

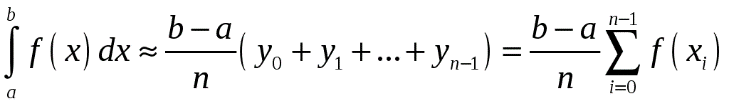
Якщо для кожної пари відрізків  побудувати многочлен другого ступеня, потім проінтегрувати його і скористатися властивістю адитивності інтеграла, то одержимо формулу Сімпсона.



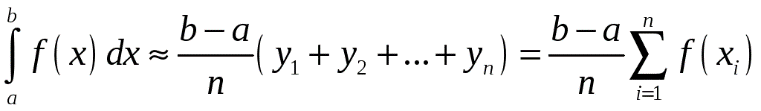
Отримане для інтеграла  значення збігається із площею криволінійної трапеції, обмеженою віссю *х*, прямими  і параболою, що проходить через точки

#### Формули прямокутників

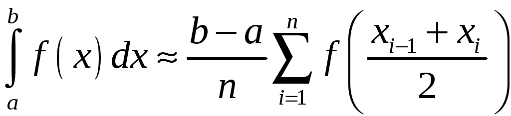
Формула лівих прямокутників



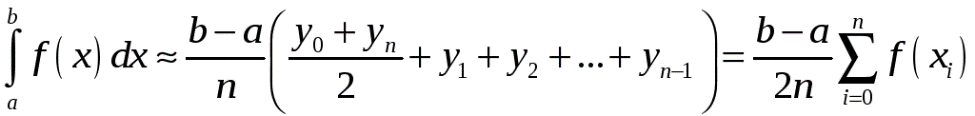
Формула правих прямокутників



Формула середніх прямокутників



#### Формула трапецій



#### Приклад коду

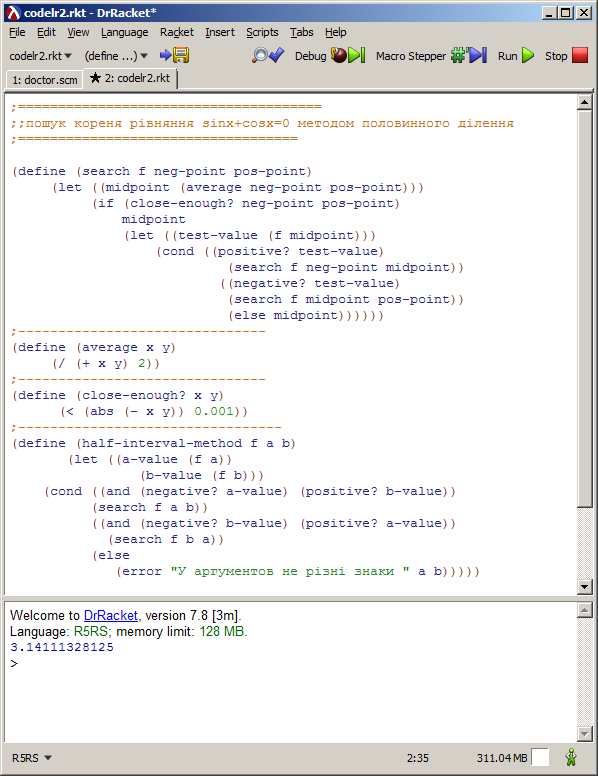


Рис. 1. Приклад коду задачі пошук кореня рівнянн методом половинного ділення

## Завдання до лабораторної роботи 3

1. **Написати процедури, що знаходять корені нелінійних рівнянь, використовуючи форми lambda, let, set!**
2. **Написати процедури, що обчислюють інтеграл функції за формулами прямокутників, трапецій, Сімпсона (парабол)**

|  |  |
| --- | --- |
| **Номер варіанта** | **Умова завдань** |
|  | 1.1 Розв’язати нелінійне рівняння *x*=cos(*x*) методами перебору та хорд, визначивши інтервал [*a, b*], на якому існує рішення рівняння. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 1.2. За допомогою формули Сімпсона інтеграл функції *f*(*х*) від *a* до *b* наближено обчислюється у вигляді:    де *h = (b - a) / n*, для якогось парного цілого числа *n*, yk = *f* (*a* + *kh*). (Збільшення *n* підвищує точність наближеного обчислення.) Визначити процедуру, яка приймає в якості аргументів *f, a, b, n,*та повертає значення інтеграла, обчисленого за формулою Сімпсона. |
|  | 2.1 Знайти корені нелінійного рівняння виду *x* = ln(*x*)+2. Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами перебору та дотичних. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 2.2 Якщо *f(х)* - функція, а *dx* - деяке мале число, то згладжена версія *f(х)* є функція, значення якої в точці *x* є середнє між *f (x - dx), f (x)* і *f (x + dx).* Напишіть процедуру, яка в якості параметрів приймає процедуру, яка обчислює *f(х)*, і повертає процедуру, яка обчислює згладжену версію *f(х).* |
|  | 3.1. Знайти корені нелінійного рівняння виду ***x*2=e-*x***. Пошук наближеного значення хоча б одного кореня рівняння ***f*(*x*) = 0** на відрізку [*a*; *b*] здійснювати методами перебору та хорд. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 3.2 Написати процедури для обчислити інтеграла за формулами прямокутників і трапецій. Порівняти результати обчислення. |
|  | 4.1 Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами дотичних та Ньютона. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 4.2 Написати процедури для обчислити інтеграла за формулами прямокутників і Сімпсона. Порівняти результати обчислення. |
|  | 5.1 Знайти корені нелінійного рівняння виду cos(*x)* – *x* + 5=0. Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами бісекції та Ньютона. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 5.2 Написати процедури для обчислити інтеграла за формулами Сімпсона і трапецій. Порівняти результати обчислення. |
|  | 6.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами хорд та простої ітерації. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 6.2 Написати процедури для обчислити інтеграла за формулами прямокутників і Сімпсона. Порівняти результати обчислення. |
|  | 7.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами Ньютона та простої ітерації. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами.  . |
| 7.2 Написати процедури для обчислити інтеграла за формулами лівих і правих прямокутників. Порівняти результати обчислення. |
|  | 8.1 Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами дотичних та бісекції. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 8.2. Написати процедури для обчислити інтеграла за формулами лівих прямокутників і трапеції. Порівняти результати обчислення. |
|  | 9.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами простої ітерації та дотичних. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 9.2. Написати процедури для обчислити інтеграла за формулами правих прямокутників і Сімпсона. Порівняти результати обчислення. |
|  | 10.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами Ньютона та перебором. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 10.2 Написати процедури для обчислити інтеграла за формулами середніх прямокутників і Сімпсона. Порівняти результати обчислення. |
|  | 11.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами простої ітерації та перебором. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 11.2 Написати процедури для обчислити інтеграла за формулами трапецій і Сімпсона. Порівняти результати обчислення. |
|  | 12.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами простої ітерації та перебором. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 12.2. Написати процедури для обчислити інтеграла за формулами трапецій і прямокутників. Порівняти результати обчислення. |
|  | 13.1 Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами Ньютона та перебором. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 13.2 Написати процедури для обчислити інтеграла за формулами прямокутників і Сімпсона. Порівняти результати обчислення. |
|  | 14.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами простої ітерації та дотичних. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 14.2. Написати процедури для обчислити інтеграла за формулами прямокутників і Сімпсона. Порівняти результати обчислення. |
|  | 15.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами хорд та перебором. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 15.2. Написати процедури для обчислити інтеграла за формулами прямокутників і трапецій. Порівняти результати обчислення. |
|  | 16.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами простої ітерації та дотичних. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 16.2. Написати процедури для обчислити інтеграла за формулами прямокутників і трапецій. Порівняти результати обчислення. |
|  | 17.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами бісекції та Ньютона. Значення *a, b* інтервалу вибрати самостійно.. |
| 17.2. Написати процедури для обчислити інтеграла за формулами Сімпсона і трапецій. Порівняти результати обчислення |
|  | 18.1 Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами дотичних та Ньютона. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 18.2. Написати процедури для обчислити інтеграла за формулами і трапецій. Порівняти результати обчислення. |
|  | 19.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами перебором та бісекції. Значення *a, b* інтервалу вибрати самостійно. Порівняти результати розв’язків двома методами. |
| 19.2. Написати процедури для обчислити інтеграла за формулами прямокутників і трапецій. Порівняти результати обчислення. |
|  | 20.1. Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння *f*(*x*) = 0 на відрізку [*a*; *b*] здійснювати методами бісекціїї та дотичних. Значення *a, b* інтервалу вибрати самостійн Порівняти результати розв’язків двома методами. |
| 20.2. Написати процедури для обчислити інтеграла за формулами Сімпсона і трапецій. Порівняти результати обчислення. |

### Зміст звіту

1. Умова завдання
2. Аналіз задачі та математичні викладки для розв’язання (у випадку математичної задачі )
3. Обґрунтування вибору мови та IDE
4. Код
5. Скріншот роботи та результату
6. Оцінка достовірності результату (перевірка правильності результату)
7. Висновок

# Лабораторна робота 4 Програмування списків мовами функціонального програмування

Deadline листопад

## Теоретичні відомості

Список. Послідовність зв’язаних між собою елементів, кожен з яких є або атомом, або списком, або парою називають у функціональному програмуванні списком (List). Списки позначаються в дужках, елементи списку розділяються пробілами.

Пара. Об’єднання двох елементів в одне ціле називається парою. Елементами пари можуть бути об’єкти допустимої у функціональному програмуванні структури, тобто атом, пара, список. Пара позначається в дужках, між елементами ставиться крапка, яка розділена з обох сторін пробілами.

Члени списку організовані в пам’яті комп’ютера у вигляді послідовностей комірок, розділених на дві частини. У першій частині комірки вказується інформація про член списку. Це може бути безпосереднє значення, якщо інформація подається атомом, або адреса комірки, де є перший елемент вкладеного об’єкту, в протилежному випадку. У другій частині комірки вказується адреса зв’язку, тобто адреса знаходження наступного члена заданого списку. Якщо член списку є останнім, то у другій частині комірки вказується символ NIL (“ніщо).

Порожній список позначається ( ).

### Процедури обробки списків

Функція car повертає голову переданого в якості аргументу списку (перший елемент списку називається головою, а залишок списку - хвостом).

Функція cdr повертає хвостову частину списку.

Функція nth повертає елемент списку по його номеру. Ця функція отримує два аргументи: перший - номер елемента в переданому в якості аргументу списку (нумерація починається з нуля); другий - сам список.

Функції first, second, third, повертають відповідно перший, другий, третій, елементи переданого в якості аргументу списку.

Функція last повертає останній елемент списку.

Функція cons будує новий список з переданих їй як аргументи голови і хвоста списку.

Функція list створює список з переданих їй як елементи аргументів. Кількість аргументів довільна.

### Стандартні процедури в мові Scheme

|  |  |
| --- | --- |
| **Процедура** | **Семантика** |
| pair? | Предикат перевірки наявності пари |
| cons | Конструктор пари |
| car | Визначення голови списку |
| cdr | Визначення другого елемента пари або хвостової частини списку |
| set-car! | Замінити перше значення списку на нове значення |
| set-cdr! | Створити пару з першого та вказаного значень |
| null? | Предикат перевірки списку на пустоту |
| list? | Предикат перевірки наявності списку |
| list | Створення списку |
| length | Довжина списку |
| append | Додавання списку |
| reverse | Інвертування списку |
| list-tail | Хвостова частина після заданої позиції |
| list-ref | Значення елемента списку на заданій позиції |
| memq | Хвостова частина, починаючи з вказаного значення = member |
| memv | Хвостова частина, починаючи з вказаного значення = member |
| member | Хвостова частина, починаючи з вказаного значення |
| assq, | Повертає пару із списку. Перший елемент пари вказаний як аргумент |
| assv, | Повертає пару із списку. Перший елемент пари вказаний як аргумент |
| assoc | Повертає пару із списку. Перший елемент пари вказаний як аргумент |
| list->vector | Перетворити список на вектор |
| vector->list | Перетворити вектор на список |
| list->string | Перетворити список у рядок |
| string->list | Перетворити рядок на список |

### Комбінації

(cadr x) = (car (cdr x))

(cdar x) = (cdr (car x))

(caar x) = (car (car x))

(cddr x) = (cdr (cdr x))

(cadar x) = (car (cdr (car x)))

### Приклади коду

;========== копія списку===========

(define (list-copy lis)

(cond ((null? lis)

'())

(else

(cons (car lis)

(list-copy (cdr lis))))))

(list 1 2 3 4 5)

(list-copy (list 1 2 3 4 5))

;=========додавання двох списків===========

(define (my-append lis1 lis2)

(cond ((null? lis1)

lis2)

(else

(cons (car lis1)

(my-append (cdr lis1) lis2)))))

(define lis1 (list 1 2 3 4 5))

(define lis2 (list 6 7 8 9 0))

(my-append lis1 lis2)

;===========інвертування списку==========================

(define (my-reverse lis)

(if (null? lis)

'()

(append (my-reverse (cdr lis))

(list (car lis)))))

(define lis1 (list 1 2 3 4 ))

(my-reverse lis1)

;==========кількість елементів списку=======================

(define (my-length lis)

(cond ((null? lis)

0)

(else

(+ 1 (my-length (cdr lis))))))

(my-length lis1)

;==============доступ до елементів списку по номеру=======================

(define (my-list-ref items n)

(if (= n 1)

(car items)

(my-list-ref (cdr items) (- n 1 ))))

(define squares (list 1 4 9 16 25))

(my-list-ref squares 1)

(my-list-ref squares 2)

(my-list-ref squares 4)

;(my-list-ref squares 10) ;error

;===================================

(display 'set-car)

(define a (list 1 2 3));?

(set-car! a 3);?

(display a)

(set-cdr! a 9);?

(define a (list 9 8 7));?

(set-cdr! a 1);?

(display a)

;==========lдоступ до елементів================

(define l (list 1 2 3 4 5))

(list-tail l 2);{3 4 5}

(list-ref l 2) ;3

(member 2 (list 1 2 3 4));(2 3 4)

(member 9 (list 1 2 3 4)) ;#f

(memv 2 (list 1 2 3 4));(2 3 4)

(memq 2 (list 1 2 3 4))

;=========================

(assoc 7 (list (list 1 2) (list 3 4) (list 5 6)))

(assq 'c (list (list 'a 'b) (list 'c 'd) (list 'e 'f)))

;=============================

(define v (vector 'A 'p 'p 'l 'e))

(vector->list v)

(define lst(list 't 'o 'm 'a 't 'o))

(list->vector lst )

## Завдання до лабораторної роботи 4

1. **Написати процедури, що створюють список, модифікують його, здійснюють пошук та упорядкування значень.**
2. **Написати процедури, що моделюють бізнес-процеси в компаніях та за допомогою різних сервісів. Бізнес-процеси подати у вигляді списків.**

|  |  |
| --- | --- |
| **Номер варіанта** | **Умова завдань** |
|  | 1.1 Створити список натуральних чисел (натуральні числа >= 1), кратних 3, задавши їх кількість. Вивести створений список. Виконати такі операції:   1. додати елементи в список на задану позицію в списку; 2. підрахувати кількість парних елементів в списку; 3. замінити усі парні значення списку на середнє арифметичне елементів списку. |
| 1.2. Написати код, що моделює роботу сортувального вузла на залізниці в процесі формування складу потягу. На сортувальному вузлі формуються потяги. Нехай існує два типи вагонів. На кожний напрямок потяг складається з вагонів одного типу. Відомі такі часові характеристики для кожного потягу: інтервали між потягами, кількість вагонів у потязі, тривалість причеплення вагону до потягу, тривалість огляду сформованого потягу, загальна тривалість простою потягу до його відправлення. Вивести на екран склад кожного потягу та сценарій виконання дій з урахування часових характеристик. |
|  | 2.1 Створити список парних натуральних чисел (натуральні числа >=1), задавши їх кількість. Вивести створений список. Виконати такі операції:   1. Додати елементи на початок списку. Кількість доданих елементів задається; 2. Здійснити пошук заданого елемента та визначення позиції його в списку, у випадку відсутності елемента вивести відповідне повідомлення; 3. підрахувати кількість елементів, значення яких в заданому діапазоні. |
| 2.2 Написати код, що моделює роботу планувальника процесів в операційні системі. Нові процеси знаходяться у вхідній черзі і очікують звільнення ресурсу — адресного простору основної пам'яті. Готові до виконання процеси розташовуються в основній пам'яті і зв'язані чергою готових процесів. Процеси в цій черзі чекають на отримання процесорного часу. Процес у стані чекання завершення операції введення/виведення знаходиться в одній з черг до пристроїв уведення/виведення. Під час виконання вказівник процесу мігрує між різними чергами під управлінням програми-планувальник, яка вирішує, який із процесів, що знаходяться в черзі готових процесів, повинен бути переданий на виконання CPU. Вивести на екран черги процесів та сценарій їх міграції. |
|  | 3.1. Створити список непарних натуральних чисел (натуральні числа >=1), задавши їх кількість. Вивести створений список. Виконати такі операції:   1. додати елементи в кінець списку. Кількість доданих елементів задається; 2. інвертувати список із заданої позиції до кінця (до заданої позиції) списку; 3. сформувати новий список із елементів, які є квадратами чисел вхідного списку. |
| 3.2 Написати код, що моделює процес здачі усних іспитів в ЗВО в режимі real time. Студенти, що допущені на іспит, сформовані у список. В процесі проходження іспиту екзаменатор викликає перших двох студентів зі списку, які отримують запитання. Поки один студент спілкується з екзаменатором, другий готовить письмову відповідь, Екзаменатор задає по два запитання, відповідь на які мають дати студенти. Після спілкування з двома студентами тривалістю m хвилин, до екзаменатора викликають наступні два студента. Якщо відповіді студентів не зараховуються, то формується новий список студентів на перездачу. Вивести на екран списки студентів, які здали іспит та список тих, кому призначена перездача. Студентів можна ідентифікувати порядковими номерами. |
|  | 4.1 Створити список факторіалів цілих чисел, задавши їх кількість. Вивести створений список. Виконати такі операції:   1. Здійснити пошук максимального та мінімального елементів списку; 2. додати нові елементи в список між максимальним і мінімальним. Кількість доданих елементів задавати; 3. Визначити суму доданих елементів. |
| 4.2 Написати код, що моделює процес обслуговування пацієнтів у сімейного лікаря. Для доступу до лікаря створений список пацієнтів, які ідентифікуються порядковими номерами. До лікаря заходить пацієнт, який може перебувати у нього певний час. Якщо лікар підозрює упацієнта коронавірус, він видає направлення в лабораторію і пацієнт уходить. Якщо стан у пацієнта важкий, лікар викликає «швидку» і вона відвозить пацієнта у лікарню. Якщо симптоми у пацєнта не викликають підозрина коронаврус, але були контакти із інфікованими, то пацієнт йде на самоізоляцію. Вивести на екран списки пацієнтів, яким рекоендована самоізоляція і вписаний лікарняний, які направлені на аналіз без лікарняного, яких відвезли і лікарню, яких не обслужили через закінчення часу прийому у лікаря. |
|  | 5.1 Створити список чисел, що є степенями двійки, задавши їх кількість. Вивести створений список. Виконати такі операції:   1. Видалити із списку усі елементи, які знаходяться парних позиціях; 2. Інвертувати список, починаючи із заданої позиції; 3. Знайти суму елементів на непарних позиціях списку. |
| 5.2 Написати код, що моделює процес обслуговування покупців в касі магазину. Сформувати список покупців (можна задати їх цифрові ідентифікатори). Відомий час обслуговування касиром кожного покупця. Визначити час перебування кожного покупця у черзі, а також номер покупця, обслуговування якого потребує найменше часу. Вивести на екран сценарій процесу обслуговування (перебування у черзі) покупців. |
| 1. можна | 6.1. Створити список трикутних чисел, задавши їх кількість. Трикутні числа складають послідовність 1, 3, 6, 10, 15, 21, 28, 36, 45, …. Формула для обчислення трикутного числа: t1 = 1; tn = n + tn−1. Вивести створений список. Виконати такі операції:   1. Створити підсписок з елементів, кратних 5; 2. Видалити із списку усі елементи, що є кратні 5; 3. Перевірити, чи дорівнює сума двох послідовних трикутних чисел повному квадрату числа. Підрахувати кількість повних квадратів чисел, що утворює список |
| 6.2 Написати код, що моделює процес управління списком справ на період (день, тиждень, місяць тощо) – сценарій Dropbox Paper, Google Tasks. Нехай існує перелік завдань, яким задані пріоритети та Deadlines. Потрібно утворити список першочергових справ враховуючи пріоритети та терміни виконання. Задачі можуть бути розбиті на підзадачі, які утворюють підсписки. Відсортувати список за термінами виконання та за пріоритетами. Продемонструвати сценарій виконання завдань, вибираючи їх зі списку, вилучаючи зі списку завдання, які вже виконані, пересуваючи завдання у списку відповідно до зміни пріоритету або терміну виконання. |
|  | 7.1. Створити список квадратних чисел, задавши їх кількість: 1, 4, 9, 16, 25, 36, 49,…. Вивести створений список. Виконати такі операції:   1. Створити підсписок з парних елементів списку квадратних чисел; 2. Підрахувати кількість елементів, що є кратні 4; 3. Перевірити, що квадратні числа є сумою двох послідовних трикутних чисел 1, 3, 6, 10, 15, 21, 28, 36, 45. Формула для обчислення трикутного числа: t1 = 1; tn = n + tn−1. |
| 7.2 Написати код, що моделює процес обслуговування клієнтів call-центра. Дзвінки клієнтів формують список (чергу), в якій вони мають порядкові номери. В залежності від кількості операторів список вхідних дзвінків розбивається на декілька списків. Дзвінок направляється до того оператора, список якого найкоротший на момент поточного дзвінка. По мірі обслуговування дзвінків вони видаляються із вхідного списку та додаються до списків операторів, які їх обслуговуватимуть. Вивести на екран сценарій обслуговування операторами дзвінків клієнтів. |
|  | 8.1 Створити список чисел, які є числами Фібоначчі, задавши їх кількість. Числа Фібоначчі складають послідовність 1, 1, 2, 3, 5, 8, …, в якій кожне наступне число дорівнює сумі двох попередніх. Вивести створений список. Виконати такі операції:   1. Визначити елемент списку, вказавши його позицію в списку; 2. Довести властивість: *f*1 + *f*2 + ... + *fn* = *fn*+2 – 1, де *fn* – позначає *n*-ий член послідовності Фібоначчі; 3. Сформувати новий список з елементів попереднього, які є простими числами |
| 8.2. Написати код, що моделює процес управління проектом за гнучкою методологією Scrum. Беклог продукту складає список задач з їх пріоритетами. Пріоритети складають ряд Фібоначчі. Планування спринту передбачає створення нового списку задач із беклогу, які будуть виконуватися протягом спринту заданої тривалості. Завдання, які вибрані для спринту, видаляються із беклогу. Після виконання спринту підсумовуються бали тих задач, які були реалізовані. Планується новий спринт. Вивести на екран сценарій управління проектом за методологією Scrum. |
|  | 9.1. Створити список кубів натуральних чисел (натуральні числа >=1), задавши їх кількість. Вивести створений список. Виконати такі операції:   1. Визначити елементи списку, які є двозначними та тризначними числами; 2. Видалити двозначні та тризначні числа зі списку; 3. Знайти суму чисел в списку, що залишилися. |
| 9.2. Написати код, що моделює систему масового обслуговування (СМО). У випадкові моменти часу надходять заявки (запити, вимоги) на обслуговування, при цьому заявки, що надійшли, обслуговуються за допомогою наявних у розпорядженні системи каналів обслуговування. Надійшовши в систему обслуговування заявка приєднується до черги вимог, які раніше надійшли. Канал обслуговування вибирає заявку з тих, що знаходяться в черзі, для того, щоб приступити до її обслуговування. Після завершення процедури обслуговування чергової вимоги канал обслуговування приступає до обслуговування наступної заявки, якщо така є в черзі. Цикл функціонування системи масового обслуговування повторюється. При цьому передбачається, що перехід системи на обслуговування чергової вимоги після завершення обслуговування попередньої вимоги відбувається миттєво, у випадкові моменти часу. Вивести на екран сценарій роботи СМО. |
|  | 10.1. Створити список випадкових цілих чисел, задавши їх кількість. Вивести створений список. Виконати такі операції:   1. Визначити елементи списку, які є простими числами; 2. Створити новий список, в якому на початку списку розташовані прості числа, потім числа складені. 3. Замінити кожне третє число в списку на значення 0 (нуль) |
| 10.2 Написати код, що моделює роботу автозаправної станції. На АЗС два однакові стовпчики, продуктивність кожної з яких відома. Перший стовпчик обслуговує 1 машину в годину, другий — 3 машини в годину. Є місце, де машини можуть очікувати обслуговування. Якщо стовпчики зайняті, то на цьому місці можуть очікувати обслуговування інші машини, але не більше двох одночасно. Черга обслуговування загальна. Як тільки один зі стовпчиків звільниться, то перша машина із черги може зайняти її місце на колонку (при цьому друга машина просувається на перше місце в черзі). Якщо з'являється третя машина, а всі місця (їх два) у черзі зайняті, то їй відмовляють в обслуговуванні, тому що стояти на дорозі забороненої. Вивести на екран сценарій роботи автозаправної станції. |
|  | 11.1. Створити список досконалих натуральних чисел, задавши максимальне значення числа (10000). Досконале число – це натуральне число, яке дорівнює сумі всіх своїх дільників. Вивести створений список. Виконати такі операції:   1. Визначити елементи списку, які є факторіалами чисел, та підрахувати їх кількість; 2. Видалити елементи списку, які є факторіалом числа; 3. Знайти елементи, остання цифра яких дорівнює n (задається користувачем). |
| 11.2 Написати код, що моделює процес обслуговування пасажирів маршрутками. Пасажири утворюють список, їх ідентифікація за порядковими номерами. У маршрутку можуть увійти k людей. На маршруті m машин. Машини прибувають на зупинки кожні n хвилин. Черга пасажирів зменшується на k людей, коли прибуває маршрутка, але протягом чекання наступної машини збільшується на p<=k людей. Вивести на екран сценарій обслуговування пасажирів маршрутками. Визначити, скільки людей перевезуть m маршруток за r циклів роботи. |
|  | 12.1. Створити список пірамідальних чисел, задавши їх кількість. Формула для *n*-го пірамідального числа:  (1, 4, 10, 35,…). Вивести створений список. Виконати такі операції:   1. Перевірити, що *n*-е пірамідальне число дорівнює сумі *n* трикутних чисел. Трикутне число визначається за формулою *t1 =*1, *tn = n* + *tn*−1, , (1, 3, 6, 10, 15,…); 2. Сформувати підсписок з чисел списку, які кратні 5; 3. Видалити зі списку усі непарні числа. |
| 12.2. Написати код, що моделює процес роботи конвеєра. На комплектуючий конвеєр збирального цеху в середньому через 10 хвилин поступають 10 деталей 1-го типу і в середньому через 40 хвилин поступають 40 деталей 2-го типу. Конвеєр складається з секцій, які вміщають по 20 деталей кожного типу. Комплектація починається тільки при наявності деталей обох типів у потрібній кількості і продовжується 20 хвилин. При нестачі деталей секція конвеєру залишається пустою. Визначити ймовірність пропуску секції, середньої довжини черг по кожному типу деталей. Вивести на екран сценарій роботи конвеєра. |
|  | 13.1 Створити список п’ятикутних чисел, задавши їх кількість. П’ятикутні числа складають послідовність 1, 5, 12, 22, 35, 51,…. Формула для обчислення *n*-го п’ятикутного числа *Pn* = 1/2×*n*×(3×*n*− 1). Вивести створений список. Виконати такі операції:   1. Замінити значення елементів на парних позиціях на їх подвоєне значення; 2. Знайти середнє арифметичне чисел списку 3. Поділити список на два рівних по кількості елементів. Значення останнього елемента першого списку та першого елемента другого списку зробити однаковими. |
| 13.2 Написати код, що моделює процес роботи вантажного аеропорту. Вантажі прибувають для відправлення в аеропорт у контейнерах із інтенсивністю два контейнери за 1 хвилину. Вантажний аеропорт не має фіксованого розкладу, а літаки відправляються по мірі їх повного завантаження. У розпорядженні є два типи літаків для перевезення вантажів: три літаки з вантажопідіймальністю 80 контейнерів і два літаки з вантажопідіймальністю 140 контейнерів. Час польоту кожного літака в середньому 3 години. Керуючий аеропортом намагається якнайчастіше використовувати літаки меншої вантажопідіймальності. Літаки, що піднімають 140 контейнерів, використовуються тільки тоді, коли інших немає в наявності. Припускається, що часом вантаження можна не враховувати. Визначити середній час очікування контейнерів із вантажами та середнє завантаження літаків обох типів. Вивести на екран сценарій роботи аеропорту. |
|  | 14.1. Створити список шестикутних чисел, задавши їх кількість. Шестикутні числа складають послідовність 1, 6, 15, 28, 45,…. Формула для обчислення *n*-го шестикутного числа *Pn* = 2×*n2* − *n*. Вивести створений список. Виконати такі операції:   1. Знайти суму парних елементів списку; 2. Замінити в списку значення, кратні 5, на значення, помножені на 10. 3. Сформувати підсписок із чисел, кратних 10. |
| 14.2. Написати код, що моделює процес роботи роздрібної торгівлі. Фірма має в місті 6 точок роздрібного продажу. Попит на товари у цих точках 10 од. товару в день. Торгові точки обслуговуються оптовим магазином. На передачу запиту торгової точки в магазин потрібно 1 день. Товари за запитом поступають з оптового магазина в торгову точку в середньому через 5 днів після одержання запиту. Оптовий магазин кожні 14 днів розміщує замовлення на фабриці. Час, протягом якого магазин одержує вантаж із фабрики, в середньому складає 90 днів. Визначити обсяг запасу товару в оптовому магазині, ймовірність невдоволеного запиту торгової точки. Вивести на екран сценарій роботи роздрібної торгівлі. |
|  | 15.1. Задати дату народження у вигляді ДД.ММ.РРРР. Створити список, елементи якого складають квадрат Піфагора, що побудований за датою народження. Алгоритм побудови квадрата Піфагора: <http://www.xsp.ru/online/pifagor/description.php>. Значення, які в квадраті Піфагора відсутні, в списку записати нулями.  Вивести створений список. Виконати такі операції:   1. Підрахувати кількість нульових елементів; 2. Знайти елементи, які містять по *n* (1,2,3,або 4 - задане користувачем) цифр; 3. Сформувати новий список з елементів з найменшою та з найбільшою кількістю цифр. |
| 15.2. Написати код, що моделює процес роботи банку. Банк має 5 кас для обслуговує клієнтів протягом робочого дня (5 годин). Інтервали часу між надходженням клієнтів складають 1 хвилину. Час обслуговування клієнтів 5 хвилин. До кожної каси формується окрема черга. Клієнт, що надійшов на обслуговування, обирає найкоротшу чергу, при цьому, якщо найкоротших черг декілька, то клієнт обирає першу. В зв’язку з тим, що керівництво банку цікавлять поточні витрати та якість послуг, що надаються клієнтам, ставиться питання про можливість зменшення кількості кас. Вивести на екран сценарій роботи кас банку та кількість кас, які залишаться в банку. |
|  | 16.1. Створити список з додатних та від’ємних чисел, задавши їх кількість. Вивести створений список. Виконати такі операції:   1. Замінити від’ємні числами їх квадратами; 2. Упорядкувати список за спаданням; 3. Визначити суму парних чисел. |
| 16.2. Написати код, що моделює процес роботи ліфта в п’ятиповерховому офісі. Люди надходять на перший поверх кожні 2 хвилини. Для кожного з інших поверхів ймовірність того, що людина прямує саме на цей поверх, дорівнює 0,25. Час, потрібний для переміщення ліфту на один поверх складає 15 секунд. Час, потрібний для завантаження та розвантаження ліфту, не враховується. На певному поверсі людина залишається протягом години. Коли людина залишає поверх (2,3,4,5), то з ймовірністю 0,7 вона прямує на перший поверх і з ймовірністю 0,1 прямує на один із інших поверхів. Місткість ліфту – 6 людей. Якщо людина не вміщується у ліфт, то вона залишається в очікуванні ліфту на своєму поверсі. На початку моделювання ліфт знаходиться на першому поверсі. Визначити: середній час очікування в усіх чергах (на кожному поверсі у двох напрямках вверх і униз); максимальну та середню кількість людей у ліфті. Вивести на екран сценарій роботи ліфта. |
|  | 17.1. Створити два списки цілих чисел. Вивести їх на екран. Виконати такі операції:   1. Побудувати третій список, в якому кожний елемент дорівнює найбільшому спільному дільнику відповідних елементів введених списків. 2. Визначити елементи, які співпадають в обох списках та надрукувати їх; 3. Видалити з першого списку елементи, які співпадають з елементами другого списку; |
| 17.2. Написати код, що моделює процес зарахування абітурієнтів на магістерську освітню програму. Формується список з k абітурієнтів, що подали заяви і призначені на іспит. Після написання іспиту формується новий список, елементами якого є пари (номер абітурієнта . екзаменаційна оцінка). Другий список сортується за екзаменаційною оцінкою. Виділено m бюджетних місць (m < k). З другого списку формується третій список, в якій переносяться перші m елементів другого списку. Ці елементи видаляються з другого списку. В процесі формування третього списку n абітурієнтів з прохідними балами забирають документи, тобто n елементів видаляються з третього списку. Для доповнення третього списку до кількості в m елементів з другого списку n елементів з позиції m+1 переносяться у список 3. Вивести на екран сценарій зарахування абітурієнтів. |
|  | 18.1 Створити список чисел або символів. Виконати такі операції:   1. Обчислити кількість елементів у найдовшій серії, вивести усі серії та шукану кількість. Серія — це послідовність однакових елементів, розташованих поспіль; 2. В кожній серії списку залишити по одному елементу та надрукувати список ; 3. Додати на початок списку n елементів, кратних заданому користувачем числу. |
| 18.2. Написати код, що моделює процес прийняття фахівця на роботу. На вакантну посаду подано k резюме, з яких сформовано список. Попередній аналіз резюме визначив пріоритети резюме. В результаті сформований новий список, в якому резюме відсортовані за пріоритетом. Вищий пріоритет має нижчий номер. На співбесіду допущено m (m<k) резюме. В процесі проходження 1 туру співбесіди n (n<m) людей отримали відмову. В процесі проходження 2 туру відмовлено p <m-n людям. Решту людей взяли на випробувальний термін. Вивести на екран сценарій модифікації списків в процесі походження 1,2 турів та випробувального періоду. Запропонувати сценарій вибору одної людини після випробувального періоду. |
|  | 19.1. Створити числовий список, кількість елементів якого задана користувачем. Значення елементів в списку можуть повторюватися. Вивести список на екран. Виконати такі операції:   1. Визначити позицію першого та останнього входження заданого числа в список. 2. Знайти елемент, значення якого повторюється найбільшу кількість разів. 3. Видалити елементи, що повторюються, залишивши в списку значення, що входять по одному разу. |
| 19.2. Написати код, що моделює процес навчання студентів ЗВО з певної дисципліни (за основу взяти свій власний досвід). Студент отримує перелік завдань для семестрового зарахування з певної дисципліни. Сформувати список з пар (ID\_завдання . кількість балів, які можна отримати). На початку моделювання сумарна кількість балів дорівнює 0. В процесі навчання студент здає завдання, накопичує семестрові бали. Формується новий список пар (ID‑завдання . кількість балів, які студент отримав), модифікується перший список, з якого вилучаються виконані та зараховані завдання. Вивести на екран сценарій модифікації списків в процесі навчання в семестрі. |
|  | 20.1. Створити цілочисловий список з додатних, від’ємних та нульових елементів, кількість елементів списку задана користувачем. Вивести список на екран. Виконати такі операції:   1. Сформувати підсписок з елементів між першим від’ємним та нульовим елементом та між першим додатним та нульовим; 2. Видалити нульові елементи списку; 3. Здійснити пошук елемента, значення якого є факторіалом числа |
| 20.2. Написати код, що моделює процес вантаження та розвантаження кораблів в порту. Кораблі надходять до гавані кожні 2 дні. У гавані є док з двома якірними стоянками та двома кранами для розвантаження кораблів. Кораблі, що надійшли у момент, коли обидві якірні стоянки зайняті, очікують у черзі з дисципліною обслуговування FIFO, Час, потрібний для розвантаження одним краном одного корабля, 1 дня. Якщо у гавані тільки один корабель, розвантаженням його займаються обидва крани і час розвантаження (час, що залишився) зменшується вдвічі. Якщо надходить ще один корабель, то один кран без затримки відправляється його розвантажувати. Інший кран продовжує розвантажувати корабель, що надійшов раніше, а час, що залишився на його обслуговування збільшується вдвічі. Створити списки, які містять дані про надходження кораблів та їх обслуговування. Визначити мінімальний, максимальний та середній часу обслуговування кораблів у гавані. Вивести на екран сценарій модифікації списків. |