**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.

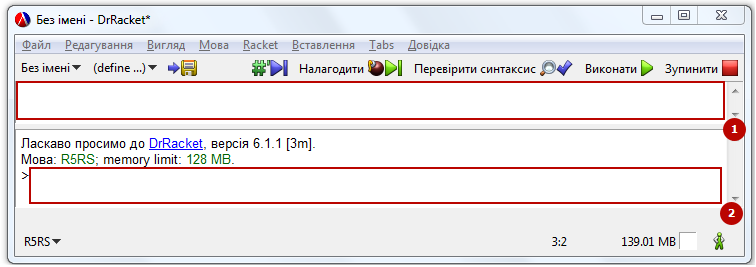


Рис. 1. Головне вікно середовища.

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).

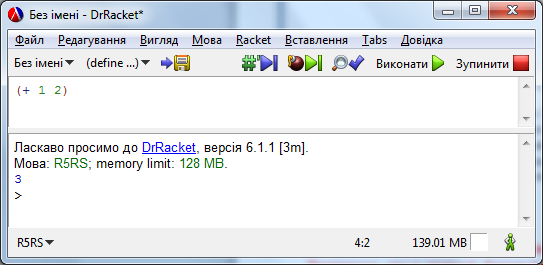


Рис. 3. Виконання програми в середовищі Dr. Racket

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).

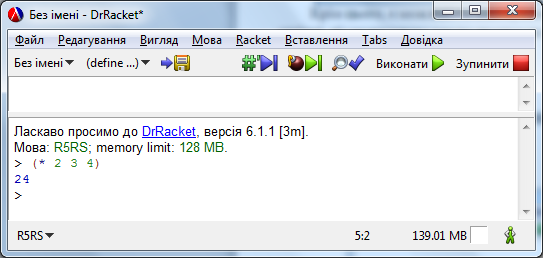


Рис. 4. Виконання програми в командному рядку

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.

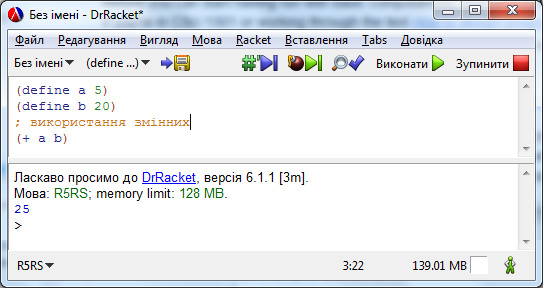


Рис. 5. Використання змінних та коментарів у середовищі Dr. Racket

### 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.

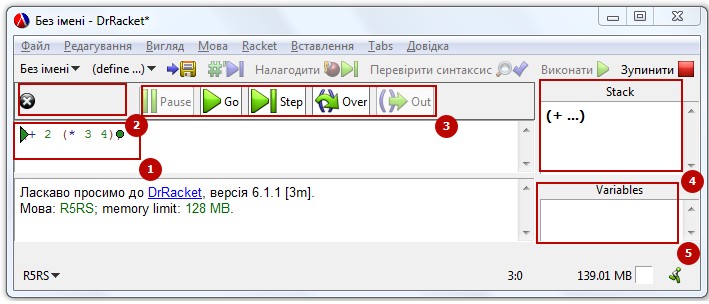


Рис. 6. Інтерфейс середовища Dr.Racket у режимі відлагодження

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.

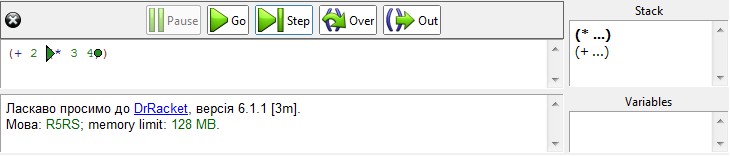


Рис. 7. Додавання в стек функції множення

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).

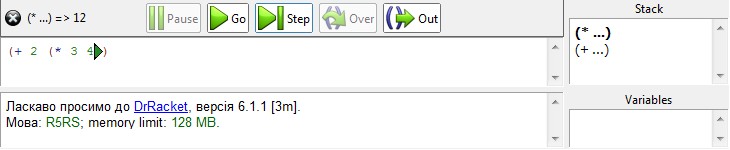


Рис. 8. Здійснення обрахунків для функції множення

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).

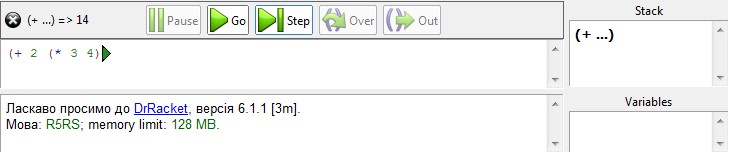


Рис. 9. Здійснення обрахунків для функції додавання

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 тригонометричнихта експоненціальних функцій

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)*.

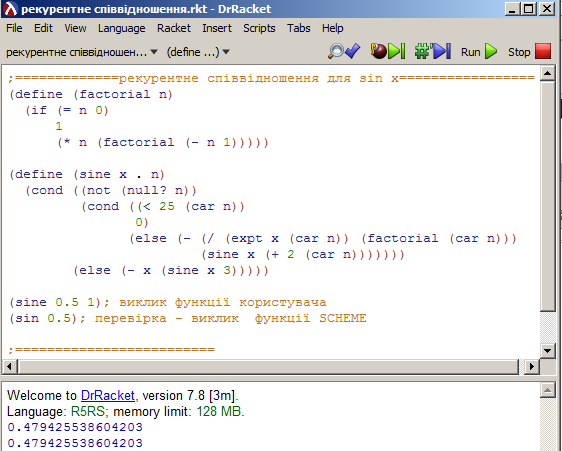
Наближене значення суми ряду можна отримати або обмежуючись сумою перших *n* його членів, або обчислюючи суму з наперед заданою точністю. Формула загального члена даного ряду є достатньо простою, але використовувати її не раціонально, оскільки для кожного члена ряду треба обчислювати степінь і факторіал. 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 |