Міністерство освіти і науки, молоді та спорту України
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
"НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ"

ТЕОРІЯ АЛГОРИТМІВ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ І ЗАВДАННЯ

до виконання лабораторних робіт з дисципліни

для студентів напряму підготовки
6.050101 Комп'ютерні науки

Дніпропетровськ
2012

Міністерство освіти і науки, молоді та спорту України
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
"НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ"

ГЕОЛОГОРОЗВІДУВАЛЬНИЙ ФАКУЛЬТЕТ

Кафедра геоінформаційних систем

ТЕОРІЯ АЛГОРИТМІВ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ І ЗАВДАННЯ

до виконання лабораторних робіт з дисципліни

для студентів напряму підготовки

6.050101 Комп'ютерні науки

Дніпропетровськ
НГУ
2012

Теорія алгоритмів. Методичні рекомендації і завдання до виконання лабораторних робіт з дисципліни для студентів напряму підготовки 6.050101 Комп'ютерні науки / Уклад.: В.О. Трусов, Є.П. Зацепін. – Д.: Національний гірничий університет, 2012. – 24 с.

Укладачі:

В.О. Трусов, канд. техн. наук, доц. (вступ, лаб. роботи 1,3,6,7),

Є.П. Зацепін, асист. (лаб. роботи 2.4,5).

Затверджено методичною комісією з напряму підготовки 6.050101 Комп'ютерні науки (протокол № 2 від 25.06.12) за поданням кафедри геоінформаційних систем (протокол № 1 від 15.06.12).

Відповідальний за випуск завідувач кафедри геоінформаційних систем Б.С. Бусигін, д-р техн. наук, проф.

ЗМІСТ

ВСТУП

Процес створення комп'ютерної програми для вирішення якого-небудь практичного завдання містить у собі формалізацію цього завдання, розробку обчислювального алгоритму її рішення, написання та налагодження програми, і, нарешті, вирішення поставленого завдання. Механізм реалізації алгоритмічного процесу зручно простежити на алгоритмічних моделях, що використовують кінцеві набори найпростіших об'єктів і елементарних дій.

Лабораторний практикум по дисципліні "Теорія алгоритмів" спрямований на вивчення основних типів алгоритмічних моделей, а також включає елементи розробки та програмування їх інтерпретації мовою високого рівня. В роботах розглянуті основні терміни та поняття теорії алгоритмів, засоби інтерпретації основних алгоритмічних моделей.

В лабораторній роботі № 1 розглянуті питання щодо основних властивостей алгоритмів на прикладі програмної реалізації типового обчислювального процесу мовою Object Pascal.

Лабораторна робота № 2 присвячена розробці алгоритмів функціонування машини Т'юринга для виконання простих функції засобами програмного інтерпретатора TMR.

В лабораторній роботі № 3 вивчаються питання реалізації композиції машин Т'юринга та роботи машини на правій напівстрічці.

У лабораторній роботі № 4 розкрито методику побудови машин Т'юринга для обчислення предикатів та реалізації умовних операторів.

Алгоритмічна модель, заснована на використанні конструктивно обумовлених арифметичних функцій, одержала назву рекурсивних функцій. Саме це питання засобами системи швидкої розробки програм Delphi розглянуто у лабораторних роботах № 5-7.

Лабораторна робота №5 присвячена обчисленню найпростіших арифметичних операцій. В роботі розглянуті відповідні схеми примітивної рекурсії та вивчаються прийоми вкладених викликів функцій і процедур засобами мови програмування високого рівня.

В лабораторній роботі № 6 присвячена розробці алгоритмів рекурсивного обчислення примітивно-рекурсивних операторів.

Лабораторна робота № 7 присвячена розробці алгоритмів обчислення функцій Акермана, що не є примітивно-рекурсивними.

Слід зазначити, що для виконання лабораторного практикуму студент повинен володіти знаннями, що отримані з лекційного курсу, а також з курсів "Вища математика" і "Основи програмування та алгоритмічні мови".

ЛАБОРАТОРНА РОБОТА №1

Побудова та програмування алгоритмів найпростіших обчислювальних процесів

Метою роботи є складання програми розкладання довільного натурального числа в добуток ступенів простих чисел. При цьому на прикладі даного алгоритму необхідно усвідомити, у чому проявляються основні властивості алгоритмів: дискретність, детермінованість, результативність, масовість.

Відомості з теорії

Основні вимоги до алгоритмів (властивості алгоритмів)

1. Перше, що слід зазначити в будь-якому алгоритмі, – це те, що він ***застосовується до вихідних даних і видає результати***. У звичних технічних термінах це означає, що алгоритм ***має входи і виходи***. Крім того, у ході роботи алгоритму з'являються ***проміжні результати***, що використовуються надалі. Таким чином, кожний алгоритм має справу з ***даними*** – вхідними, проміжними і вихідними. Оскільки ми збираємося уточнювати поняття алгоритму, потрібно уточнити також поняття даних, тобто вказати, яким вимогам повинні задовольняти об'єкти, щоб алгоритми могли з ними працювати. Ясно, що ці об'єкти повинні бути чітко визначеними і відмінними як друг від друга, так і від "не-об'єктів". У багатьох важливих випадках добре зрозуміло, що це значить: до таких алгоритмічних об'єктів належать числа, числові масиви (вектори, матриці), формули. Зображення (наприклад, географічна карта) представляється менш природним як алгоритмічний об'єкт: малюнок набагато складніше розбити на елементи, ніж числовий масив. Нарешті, з такими об'єктами, як "гарна книга" або "осмислене твердження", з якими легко управляється будь-яка людина (але кожен по-своєму!), алгоритм працювати відмовиться, поки вони не будуть описані як дані за допомогою інших, більш придатних об'єктів.

Набір елементарних об'єктів утворить кінцевий ***алфавіт*** вихідних символів (цифр, букв і т.д.), з яких будуються інші об'єкти; типовим засобом побудови є індуктивні визначення, що вказують, як будувати нові об'єкти із уже побудованих. Найпростіше індуктивне визначення – це визначення деякої множини слів, класичним прикладом якого служить визначення ідентифікатора в мові Pascal: ідентифікатор – це або буква, або ідентифікатор, до якого приписана праворуч буква або цифра. Слова кінцевої довжини в кінцевих алфавітах (зокрема, числа) – найбільш звичайний тип алгоритмічних даних, а число символів у слові (довжина слова) – природна одиниця виміру об'єму оброблюваної інформації. Більш складний випадок алгоритмічних об'єктів – формули. Вони також визначаються індуктивно і також є словами в кінцевому алфавіті, однак не кожне слово в цьому алфавіті є формулою.

2. ***Дані для свого розміщення вимагають пам'яті***. Пам'ять звичайно вважається однорідною і дискретною, тобто складається з однакових комірок, причому кожна комірка може містити один символ алфавіту даних. Таким чином, одиниці виміру обсягу даних і пам'яті погоджені. При цьому пам'ять може бути нескінченною. Питання про те, чи потрібна одна пам'ять або декілька і, зокрема, чи потрібна окрема пам'ять для кожного із трьох видів даних (вхідних, вихідних і проміжних), вирішується по-різному.

3. ***Алгоритм складається з окремих елементарних кроків, дій, причому множина різних кроків, з яких складений алгоритм, є кінцевою.*** Типовий приклад множини елементарних дій – система команд ЕОМ. Звичайно елементарний крок має справу з фіксованим числом символів (це зручно, наприклад, для виміру часу роботи алгоритму числом пророблених кроків). Кожна дія повинне бути закінчена перш, ніж відбудеться перехід до виконання наступної дії. Це властивість алгоритму називається ***дискретністю***.

4. ***Послідовність кроків алгоритму детермінована,*** тобто після кожного кроку або вказується, який крок робити далі, або дається команда зупинки, після чого робота алгоритму вважається закінченою. Завдяки цій властивості виконання алгоритму носить механічний характер і не вимагає ніяких додаткових вказівок або відомостей про розв'язувану задачу.

5. Природно від алгоритму зажадати ***результативності,*** тобто зупинки після кінцевого числа кроків (що залежить від даних) із вказівкою того, що вважати результатом. Зокрема, усякий, хто пред'являє алгоритм рішення деякого завдання, наприклад обчислення функції *f(x)*, зобов'язаний показати, що алгоритм зупиняється після кінцевого числа кроків (як кажуть, *сходиться*) для будь-якого *x* з області завдання *f*. Однак перевірити результативність (збіжність) набагато складніше, ніж вимоги, викладені в пп. 1-4. На відміну від них збіжність звичайно вдається встановити простим переглядом опису алгоритму; загального ж методу перевірки збіжності, придатного для будь-якого алгоритму А и будь-яких даних *x,* як буде показано далі, взагалі не існує.

6. ***Масовість*** алгоритму – властивість алгоритму бути застосовним для множини вхідних значень. Алгоритм вирішення задачі розробляється в загальному виді і повинен застосовуватися для деякого класу завдань, що різняться лише вихідними даними.

Завдання

Скласти програму мовою Паскаль, що здійснює розкладання натурального числа типу integer, відмінного від одиниці, у добуток ступенів простих чисел. Вихідне натуральне число повинне вводитися із клавіатури, результат виконання програми повинен виводитися на екран. Знак добутку можна позначити за допомогою "*", знак зведення в ступінь – за допомогою "^". Наприклад, запис $2^2 \cdot 3^5 \cdot 7$ повинен бути представленим у вигляді 2^2*3^5*7^1.

Складанню основної програми повинне передувати складання допоміжної функції перевірки простоти числа. Програму реалізувати в середовищі Delphi у вигляді додатка.

Результатом виконання роботи є звіт – текстовий документ, що містить тексти основної програми та функції (з коментарями) і скріншоти роботи програми для 4-5 багатозначних чисел. Після налагодження проекту зберегти проект і виконуючий файл програми разом із самим звітом у папці "Звіт по ЛР1" і пред'явити їх викладачеві.

При захисті лабораторної роботи на прикладі побудованої програми необхідно відповісти на наступні питання.

1. Яке значення понять "однорідна і дискретна пам'ять"?
2. Що таке алфавіт даних, слово, довжина слова?
3. Яка властивість алгоритму називається дискретністю?
4. Яка властивість алгоритму називається детермінованістю?
5. У чому складається результативність алгоритму?
6. Чи існує загальний метод, що дозволяє для будь-якого алгоритму перевірити його результативність?
7. У чому полягає масовість алгоритму?

ЛАБОРАТОРНА РОБОТА №2

Найпростіші алгоритми для машини Т'юринга

Метою роботи є формування знань і вмінь для складання програм для машини Т'юринга.

У процесі виконання роботи формуються наступні вміння:

- складання алгоритмів для рішення простих задач за допомогою машини Т'юринга;
- кодування алгоритмів і їхня реалізація в Програмній системі моделювання роботи машини Т'юринга.

Відомості з теорії

Машина Т'юринга складається з:

1) керуючого пристрою. Його стани утворюють кінцеву множину $Q = \{q_1,...,q_n\}$;

2) стрічки, розбитої на комірки, у кожній з яких може бути записаний один із символів кінцевого алфавіту $A = \{a_1,...,a_m\}$;

3) пристрою звертання до стрічки, тобто голівки, що зчитує і пише. У кожний момент часу голівка займає одну комірку стрічки. Залежно від символу у цій комірці і стану керуючого пристрою голівка записує в комірку символ (бути може такий, що збігається з попереднім, або порожній, тобто стирає символ), зрушується на комірку уліво або вправо або залишається на місці; при

цьому керуючий пристрій переходить у новий стан (або залишається в старому);

4) системи команд.

Серед станів керуючого пристрою виділені початковий стан q_1 і заключний стан, що будемо позначати q_z (z тут розуміється не як числова змінна, а як мнемонічний знак кінця). У початковому стані машина перебуває перед початком роботи. Потрапивши в заключний стан, машина зупиняється.

Таким чином, пам'ять машини Т'юринга – це кінцева безліч станів (внутрішня пам'ять) і стрічка (зовнішня пам'ять). Стрічка нескінченна в обидва боки, однак у початковий момент часу тільки кінцеве число комірок стрічки заповнено непустими символами, інші комірки порожні, тобто містять порожній символ λ (пробіл).

Дані машини Т'юринга – ц, зрушення голівки на комірку уліво і вправо, а також перехід керуючого пристрою в наступний стан. Детермінованість машини, тобто, послідовність її кроків, визначається в такий спосіб: для будь-якого внутрішнього стану q_i і символу е слова в алфавіті стрічки; на стрічці записуються як вихідні дані, так і остаточні результати. *Елементарні кроки машини* – це зчитування та запис символів a_j однозначно задані:

а) наступний стан $q_i^{/}$;

б) символ $a_j^{/}$, що потрібно записати замість a_j у ту ж комірку (стирання символу будемо розуміти як запис порожнього символу λ);

в) напрямок зрушення голівки d, що позначається одним із трьох символів: *L* (уліво), *R* (вправо), *E* (на місці). Це завдання може описуватися або системою правил (команд), що мають вигляд

$$q_i a_j \to q_i^{/} a_j^{/} d, \qquad (1)$$

або таблицею, рядкам якої відповідають стани, стовпцям – вхідні символи, а на перетинанні рядка q_i, і стовпця a_j записана трійка символів $q_i^{/} a_j^{/} d$ і, нарешті, блок-схемою, що будемо називати діаграмою переходів. У цій діаграмі станам відповідають вершини, а правилу вигляду (1) – дуга, що веде з q_i в $q_i^{/}$, на якій написане $a_j \to a_j^{/} d$.

Положення машини Т'юринга, за яким однозначно можна визначити її подальшу поведінку, визначається її внутрішнім станом, станом стрічки (тобто словом, записаним на стрічці) і позицією голівки на стрічці. Це положення будемо називати *конфігурацією,* або машинним словом, і позначати трійкою $\alpha_1 q_i \alpha_2$, де q_i – поточний внутрішній стан, α_1 – слово зліва від голівки, а α_2 – слово, що створене символом, позначеним голівкою, та символами справа від нього, причому зліва від α_1 та справа від α_2 немає не порожніх символів. Наприклад, конфігурація із внутрішнім станом q_i, у якій на стрічці записано $abcde$, а голівка вказує на символ d, запишеться як $abcq_ide$. *Стандартною початковою* конфігурацією назвемо конфігурацію виду $q_1\alpha$, тобто конфігурацію, що містить початковий стан, у якій голівка вказує на крайній

лівий символ слова, написаного на стрічці. Аналогічно *стандартною заключною конфігурацією* назвемо конфігурацію виду $q_z\alpha$. Якщо до не заключної конфігурації K машини T застосовна команда виду (1), що K переводить у конфігурацію K', то таке відношення між конфігураціями позначимо $K \underset{T}{\rightarrow} K'$; якщо з контексту ясно, про яку машину T мова йде, індекс T будемо опускати. Якщо для K_1 та K_n існує послідовність конфігурацій $K_1, K_2, ..., K_n$ така, що

$$K_1 \underset{T}{\rightarrow} K_2 \underset{T}{\rightarrow} ... \underset{T}{\rightarrow} K_n,$$

то позначимо це $K_1 \underset{T}{\Rightarrow} K_n$. Наприклад, якщо в системі команд машини T є команди $q_2a_5 \rightarrow q_3a_4R$ і $q_3a_1L \rightarrow q_4a_2$, то $q_2a_5a_1a_2 \rightarrow a_4q_3a_1a_2 \rightarrow q_4a_4a_2a_2$ і, отже, $q_2a_5a_1a_2 \Rightarrow q_4a_4a_2a_2$. Послідовність конфігурацій $K_1 \rightarrow K_2 \rightarrow K_3 \rightarrow ...$ однозначно визначається вихідною конфігурацією K_1 і повністю описує роботу машини T, починаючи з K_1.

Може виявитися, що в системі команд відсутня команда, з лівою частиною qa для деякого не кінцевого стану q і деякого символу a. Якщо в процесі роботи машина переходить у конфігурацію виду $\alpha qa\beta$, то будемо вважати, що машина завершила роботу аварійною. Насправді немає істотної різниці між аварійним завершенням роботи машини Т'юринга і зацикленням: доповнюючи існуючу систему команд ні до чого не зобов'язуючою командою $qa \rightarrow qa$, ми отримаємо машину, що, досягши конфігурації $\alpha qa\beta$, зациклиться. Тому в літературі, як правило, розглядаються машини без аварійного завершення роботи. Із цього погляду послідовність конфігурацій $K_1 \rightarrow K_2 \rightarrow K_3 \rightarrow ...$ кінцева, якщо в ній зустрінеться заключна конфігурація і нескінченна в противному випадку.

Найпростішим прикладом машини Т'юринга є машина з алфавітом $A=\{1,\lambda\}$, станами $\{q_1, q_z\}$ і системою команд $q_11 \rightarrow q_11R$, $q_1\lambda \rightarrow q_11R$, що із будь-якої початкової конфігурації буде працювати нескінченно, заповнюючи одиницями всю стрічку вправо від початкової точки.

Якщо $\alpha_1q_1\alpha_2 \underset{T}{\Rightarrow} \beta_1q_z\beta$, будемо говорити, що машина T переробляє слово $\alpha_1\alpha_2$ у слово $\beta_1\beta_2$, і позначати це $T(\alpha_1\alpha_2)=\beta_1\beta_2$.

Нехай f – функція, що відображає множину векторів над $A_{исх}$ у множину векторів над $A_{рез}$. Машина T *обчислює* функцію f, якщо:

1) для будь-яких V і W, таких, що $f(V)=W$, $q_1V^* \underset{T}{\Rightarrow} q_zV^*$, де V^* і W^* – вірні записи V і W відповідно;

2) для будь-якого V, такого, що $f(V)$ не визначено, машина Т, запущена в стандартній початковій конфігурації q_1V^*, працює нескінченно.

Якщо для функції f існує машина Т, що неї обчислює, то функція f називається обчислюваною *за Т'юрингом.*

З іншого боку, усякій обчислюваній машині Т'юринга, тобто машині, що, почавши зі стандартної початкової конфігурації $q_1\alpha$, може зупинитися тільки в стандартній заключній конфігурації $q_z\beta$, можна поставити у відповідність функцію, що обчислюється їй.

Машини Т'юринга з однаковим алфавітом $A_{исх}$ будемо називати *еквівалентними,* якщо вони обчислюють ту саму функцію.

Завдання

Установіть програму "Машина Т'юринга" у своїй папці на мережевому диску, для чого з папки ТА скопіюйте файл tmr_install у свою мережеву папку і запустіть його. Використовуючи довідкову систему, ознайомтеся з "Програмною системою моделювання роботи машини Т'юринга" (система TMR). Ознайомтеся з наявними програмами.

В TMR реалізувати наступні алгоритми:

1. Інверсія двійкового слова (розв'язання розглянуте нижче).
2. Додавання вправо (алгоритм розглянутий на лекції).
3. Додавання вліво (скласти алгоритм і програму самостійно).
4. Додавання за допомогою переносу першого складнику справа від другого (алгоритм розглянутий на лекції).
5. Копіювання (алгоритм розглянутий на лекції).
6. Десяткове додавання $n+m$ (алгоритм розглянутий на лекції).
7. Десяткове вирахування $n-m$ (скласти алгоритм і програму самостійно).

У якості прикладу розглянемо кодування програми **інверсії** двійкового числа (замінити всі 0 на 1 і всі 1 на 0) у системі TMR.

Система команд для розв'язання задачі:

$q_1 1 \rightarrow q_1 0R$, $q_2 0 \rightarrow q_2 0L$,
$q_1 0 \rightarrow q_1 1R$, $q_2 1 \rightarrow q_2 1L$,
$q_1 \lambda \rightarrow q_2 \lambda L$, $q_2 \lambda \rightarrow q_z \lambda R$.

В TMR порожній символ λ звичайно позначають "_". Запис програми ведеться у вигляді таблиці переходів (рис. 1). Щоб додати символи алфавіту, у таблицю переходів необхідно вставити нові стовпці. Кожний з них відповідає одному символу алфавіту. Додавання стовпців і рядків виконується з контекстного меню або відповідною кнопкою панелі інструментів TMR. Для нашої задачі необхідно три стовпчики – "_", "0", "1".

В алфавіт машини (вікно праворуч) нові символи будуть включені автоматично.

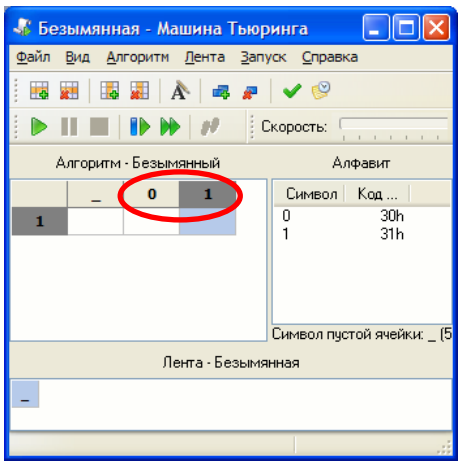


Рис. 1. Таблиця переходів системи TMR

Кожний рядок таблиці відповідає стану машини. У наведених вище командах є два стани q_1, q_2 і особливий кінцевий стан – q_z. У системі TMR кінцевому стану q_z відповідає нульовий стан – символ 0. Кожний стан (крім кінцевого) – це окремий рядок таблиці переходів. Оскільки нам необхідно два стани, додамо ще один рядок 2. У вихідній системі команд стани записувалися як q_1 і q_2, а в TMR їм будуть відповідати стани 1 і 2 (рядки 1 і 2 таблиці, див. рис. 2).

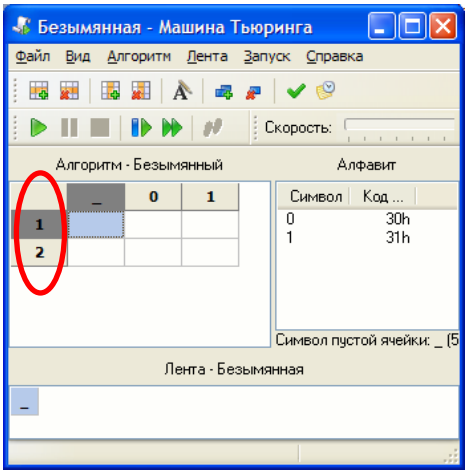


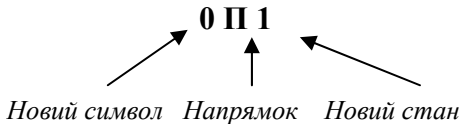
Рис. 2. Стани машини Т'юринга у системі TMR

Тепер система готова до безпосереднього уведення команд. Розглянемо порядок уведення команд.

Уведення першої команди: $q_1 1 \rightarrow q_1 0R$. Ліва частина виразу – поточний стан. Це стан q_1 і символ на стрічці "1". У системі TMR стану q_1 відповідає перший рядок, а символу на стрічці – стовпець із заголовком "1". В осередок на перетинанні цього рядка і стовпця і будемо записувати команду. Права частина виразу показує новий стан (q_1), новий символ стрічки ("0") і напрямок руху голівки вправо (R). В TMR формат запису небагато відрізняється, команди записуються в такому вигляді:

< Новий символ > < Переміщення > < Новий стан >

Праву частину команди $\mathbf{q_1 0 R}$ перетворимо в $\mathbf{0\ R\ q_1}$. Необхідно також урахувати, що запис станів в TMR відрізняється, і, як говорилося вище, $\mathbf{q_1}$ записується як стан 1. Крім того, напрямок руху вправо – це символ **П.** Тоді команда прийме вигляд:



Саме це значення і уводимо в комірку, розділяючи між собою символи пробілом (рис. 3).

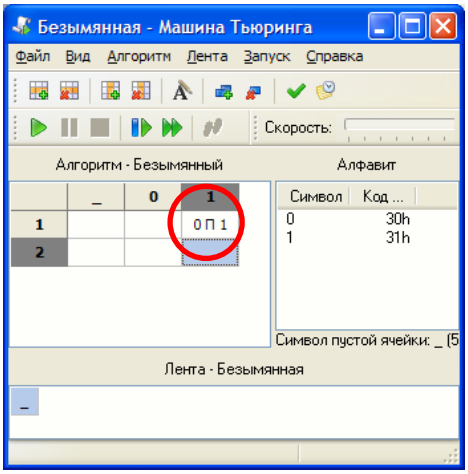


Рис. 3. Команди машини Т'юринга у системі TMR

Аналогічно введемо інші п'ять команд, перетворивши їх до формату TMR.

У нижній частині вікна введемо, наприклад, символи стрічки 111101 (використовуйте контекстне меню для комірок стрічки). Машина готова до запуску. Перед запуском не забудьте зберегти дані – програму, алгоритм, стрічку.

Машина може працювати у двох режимах – автоматичному (команди виконуються послідовно із заданою швидкістю), ручному – після виконання кожної команди машина зупиняється і чекає до запуску наступного кроку. При запуску виконується перевірка на можливі помилки і наявність заключного стану.

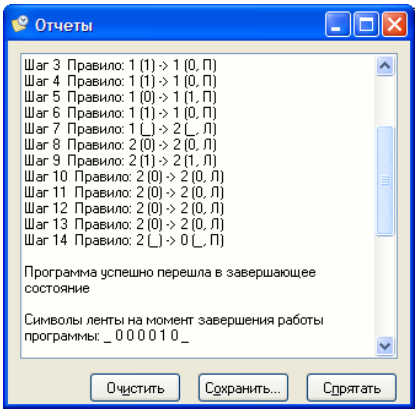


Рис. 4. Звіт про виконані команди та отримані результати

Після переходу в завершальний стан виводиться звіт про виконані команди та отримані результати (рис. 4).

Варіанти для завдань 2, 3, 4:

Перший доданок *x* = ((номер комп'ютера+13) mod 7)+3,

другий доданок *y* = ((номер студента в журналі+8) mod 6)+2,

де **mod** – операція обчислення залишку від ділення.

Перше число для **завдань 6 і 7** (десяткове додавання і вирахування) ***x*** = (номер комп'ютера+13)*4.

Результатом виконання лабораторної роботи є текстовий документ, що містить для кожного із сімох завдань:

- програму у вигляді набору команд $q_i a_j \to q_i^{/} a_j^{/} d$,
- відповідний запис алгоритму у вигляді таблиці переходів (можна використовувати скріншоти екрана з TMR) і у вигляді діаграми переходів (побудувати граф за набором команд засобами малювання Word),
- приклади вихідних стрічок і результатів,
- журнали виконання команд (скріншоти вікна звітів).

У своєму каталозі зберігайте файли TMR – алгоритм, стрічку, програму для кожного завдання в папці лабораторної роботи 2.

При захисті лабораторної роботи необхідно **відповісти на наступні питання**.

1. З яких частин складається машина Т'юринга?
2. Що означають вирази $K \to K_1$ і $K \Rightarrow K_1$?
3. Які дії виконує машина Т'юринга по команді $q_3 * \to q_{10} 0L$?
4. Що означає вираз $T(\alpha) = \beta$?
5. У якому випадку кажуть, що машина Т'юринга T обчислює функцію f?
6. Яка функція називається обчислюваною за Т'юрингом?
7. Які дві машини Т'юринга називаються еквівалентними? Є чи в даній лабораторній роботі еквівалентні машини Т'юринга?

ЛАБОРАТОРНА РОБОТА №3

Моделювання роботи композиції машин Т'юринга

Метою роботи є закріплення знань і формування вмінь для складання програм композицій машин Т'юринга.

У процесі виконання роботи формуються наступні вміння:

- складання алгоритмів, що моделюють композиції машин Т'юринга;
- кодування алгоритмів і їхня реалізація в Програмній системі моделювання роботи машини Т'юринга.

Відомості з теорії

Композицією двох функцій $f_1(x)$ і $f_2(y)$ називається функція $g(x) = f_2(f_1(x))$, що утворюється при застосуванні f_2 до результату обчислення f_1. Для того, щоб $g(x)$ була визначена при даному x, необхідне і достатньо, щоб f_1 була визначена на x и f_2 була визначена на $f_1(x)$.

Теорема. Якщо $f_1(x)$ і $f_2(y)$ обчислювані за Т'юрингом, то їхня композиція $f_2(f_1(x))$ також обчислювана за Т'юрингом.

Виявляється, що для обчислення на машині Т'юринга досить, щоб стрічка була нескінченною тільки в один бік, наприклад, вправо. Така стрічка називається правою напівстрічкою.

Теорема. Будь-яка функція, обчислювана за Т'юрингом, обчислювана на машині Т'юринга із правою напівстрічкою; інакше кажучи, для будь-якої машини Т'юринга T існує еквівалентна їй машина T_R із правою напівстрічкою. Аналогічна теорема формулюється для лівої напівстрічки.

Ідеї доказу. Можна припустити, принаймні, дві ідеї побудови такої еквівалентної машини для роботи на правій напівстрічці:

1) щораз, коли голівці колишньої машини треба зайти за лівий край стрічки, нова машина попередньо зрушить все написане (разом з голівкою) на клітинку вправо;

2) стрічка "перегинається навпіл"; при цьому комірки правої половини колишньої стрічки розміщаються, скажемо, у непарних комірках напівстрічки, а комірки лівої половини – у парні.

Завдання 1

Відповідно до варіанта (див. нижче) реалізувати систему команд композиції машин Тюрінга $T_2(T_1)$.

Номер варіанта визначається як (N mod 14) + 1, де N – номер у журналі.

Таблиця 3.1

Номер варіанта	T_1	T_2
1	$T_{коп}$	T_+
2	T_+	$T_{коп}$
3	T_{++}	$T_{коп}$
4	Машина, що пише на стрічку слово TAMOPZ	Машина с алфавітом $\{T, A, M\}$, що стирає два перших символи
5	Машина, що реалізує функцію $y = x + 2$	$T_{коп}$
6	Машина, що додає в кінець слова з одиниць слово *11	T_+
7	Машина, що додає в кінець слова з одиниць слово *1*1	T_{++}
8	Машина, що реалізує функцію $y = x + 1$	Машина, що реалізує функцію $y = x + 2$
9	T_+	Машина, що реалізує функцію $y = x + 1$
10	Машина, що реалізує функцію $y = x + 3$	$T_{коп}$
11	Машина, що пише на стрічку слово 1111	$T_{коп}$
12	Машина, що пише на стрічку слово 111*1*1	T_{++}
13	Машина, що пише на стрічку слово 111*1	T_+
14	$T_{коп}$	$T_{коп}$

Завдання 2

Побудувати машину Т'юринга, що виконує зрушення вправо на одну комірку для вхідного слова $|^a * |^b * |^c$, де a=(N mod 5) + 1, b=2, c=3.

Вимоги до оформлення див. у завданні до лабораторної роботи №2.

При захисті лабораторної роботи необхідно **відповісти на наступні питання**.

1. Яка машина Т'юринга називається композицією машин T_1 і T_2?
2. Як будується система команд композиції машин T_1 і T_2?
3. Сформулюйте теорему про композицію машин T_1 і T_2.
4. Відобразіть композицію 3-х машин Т'юринга за допомогою блок-схеми.
5. Як створити машину Тюрінга, що працює на правій (лівій) напівстрічці?
6. Запишіть результат $T_{коп}(T_+)$ при вхідному слові 11*111.
7. Запишіть результат $T_+(T_{коп})$ при вхідному слові 1111.

ЛАБОРАТОРНА РОБОТА №4

Машини Т'юринга для обчислення предикатів

Метою роботи є закріплення знань і формування вмінь для складання програм обчислення предикатів машиною Т'юринга.

У процесі виконання роботи формуються наступні вміння:

- складання алгоритмів, що моделюють машини Т'юринга для обчислення предиката подільності;
- складання алгоритмів, що моделюють машини Т'юринга для обчислення предиката порівняння;
- складання алгоритмів, що моделюють машини Т'юринга для виконання умовного переходу;
- кодування алгоритмів і їхня реалізація в Програмній системі моделювання роботи машин Т'юринга.

Відомості з теорії

Машина Т обчислює предикат $P(\alpha)$ (α – слово в $A_{вих}$), якщо $T(\alpha)=\omega$, де $\omega = И$, коли $P(\alpha)$ істинне і $\omega = Л$, коли $P(\alpha)$ хибне. Якщо ж $P(\alpha)$ не визначений, то машина Т, як і при обчисленні функцій, не зупиняється. При звичайному обчисленні предиката знищується α, що може виявитися незручним, якщо після Т повинна працювати інша машина. Тому введемо поняття обчислення з відновленням: машина Т обчислює $P(\alpha)$ з відновленням, якщо $T(\alpha)=\omega\alpha$. Якщо існує машина Т, що обчислює $P(\alpha)$, то існує і $\widetilde{T}$, що обчислює $P(\alpha)$ з відновленням. Дійсно, обчислення з відновленням можна представити наступною послідовністю конфігурацій:

$$q_1\alpha \Rightarrow q_{n_1}\alpha * \alpha \Rightarrow \alpha * q_{n_2}\alpha \Rightarrow \alpha * q_{n_3}\omega \Rightarrow q_{n_4}\omega\alpha .$$

Перша частина цієї послідовності реалізується машиною $T_{коп}$, друга – простим зрушенням голівки до маркера "*", третя – машиною T_R, що обчислює $P(\alpha)$ на правій напівстрічці, четверта – переносом ω у крайнє ліве положення. Машина $\widetilde{T}$ є композицією зазначених чотирьох машин. Однак у конкретних випадках можливі і більше прості способи відновлення α.

Нехай функція $f(\alpha)$ задана описом "якщо $P(\alpha)$ істинне, то $f(\alpha)=g_1(\alpha)$", інакше $f(\alpha)=g_2(\alpha)$ (під "інакше" мається на увазі "якщо $P(\alpha)$ хибне"); якщо ж $P(\alpha)$ не визначений, то $f(\alpha)$ також не визначена. Функція $f(\alpha)$ називається *умовним переходом до* $g_1(\alpha)$ і $g_2(\alpha)$ за умовою $P(\alpha)$ або розгалуженням g_1 і g_2 по P.

Теорема. Якщо $g_1(\alpha)$, $g_2(\alpha)$ і $P(\alpha)$ обчислювані за Т'юрингом, то розгалуження g_1 і g_2 по P також обчислюване.

Доказ. Нехай T_1 – машина з початковим станом q_1, кінцевим станом q_{1z} і системою команд Σ_1, що обчислює g_1; T_2 – машина з початковим станом q_2, кінцевим станом q_{2z} і системою команд Σ_2, що обчислює g_2; T_P обчислює з відновленням $P(\alpha)$, q_{3z} – її кінцевий стан. Тоді машина T, що обчислює розгалуження g_1 і g_2 по P, – це композиція T_P і машини T_3, система команд Σ_3 якої має такий вигляд:

$$\Sigma_3 = \Sigma_1 \cup \Sigma_2 \cup \{q_{3z}И \to \lambda\, q_{11}R, \quad q_{3z}Л \to \lambda\, q_{21}R, \quad q_{1z} \to q_{2z}E\},$$

де q_{3z} – заключний стан машини T_P.

Перші дві з нових команд передають керування системам команд Σ_1 або Σ_2 залежно від значення предиката $P(\alpha)$. Третя команда уведена для того, щоб T_3 мала один заключний стан q_{2z}. Відсутність символу стрічки в цій команді означає, що вона виконується при будь-якому символі. Теорему доведено.

Завдання 1 і 2

Відповідно до варіанта (див. табл. 4.1) реалізувати систему команд для предикатів, що обчислюються. Предикат подільності (зад. 1) обчислити без відновлення.

Номер варіанта визначається як (N mod 14) + 1, де N – номер у журналі. Значення x і y вибрати самостійно. На прикладах різних значень x, y одержати значення И і Л предикатів, що обчислюються.

Таблиця 4.1

Варіант	Зад. 1. Предикат подільності	Зад. 2. Предикат порівняння
1	«x парне»	«x більше y» з відновленням
2	«x ділиться на 3»	«x менше y» без відновлення
3	«x ділиться на 5»	«x менше або дорівнює y» з відновленням
4	«x непарне»	«x не дорівнює y» без відновлення
5	«x ділиться на 7»	«x менше або дорівнює y» без відновлення
6	«x парне»	«x менше y» з відновленням
7	«x не ділиться на 3»	«x більше або дорівнює y» без відновлення
8	«x не ділиться на 5»	«x не дорівнює y» з відновленням
9	«x не ділиться на 7»	«x менше або дорівнює y» без відновлення
10	«x ділиться на 3»	«x менше y» з відновленням
11	«x непарне»	«x менше або дорівнює y» з відновленням
12	«x ділиться на 7»	«x більше y» без відновлення
13	«x не ділиться на 5»	«x не дорівнює y» без відновлення
14	«x не ділиться на 3»	«x більше y» з відновленням

Завдання 3

Реалізувати систему команд для виконання умовного переходу за предикатом P(x) (свій варіант за завданням 1, але ***з відновленням***) до функцій g_1 і g_2 (див. табл. 4.2).

Таблиця 4.2

Варіант	g_1	g_2
1	$y = x+1$	$y = x+2$
2	$y = x+2$	$y = x+3$
3	$y = x+2$	$y = x+1$
4	$y = x$	$y = x+3$
5	$y = x+2$	$y = x$
6	$y = x+1$	$y = x$
7	$y = x+3$	$y = x+2$
8	$y = x+2$	$y = x+1$
9	$y = x+2$	$y = x+3$
10	$y = x+1$	$y = x+4$
11	$y = x$	$y = x+2$
12	$y = x+2$	$y = x+1$
13	$y = x+2$	$y = x+4$
14	$y = x+4$	$y = x$

Вимоги до оформлення звітів див. у завданні до лабораторної роботи №2.

При захисті лабораторної роботи необхідно відповісти на наступні питання.

1. Дайте визначення деякого предиката для обчислення машиною Т'юринга.

2. У чому складається обчислюваність за допомогою машини Т'юринга деякого предиката P з відновленням?

3. Відобразіть обчислення деякого предикату з відновленням за допомогою композиції машин Тюрінга.

4. Запишіть систему команд МТ $T_{\text{отр}}$, що обчислює заперечення, тобто $T(И) = Л,\ T(Л) = И$. Зобразіть граф переходів.

5. Як на машині Тюрінга реалізувати обчислення двох функцій за умовою?

6. Зобразіть граф переходів машини багаторазового додавання T_{++}

7. Зобразіть граф переходів машини визначення непарності числа в унарному коді $T_{\text{нечет}}$ без відновлення.

ЛАБОРАТОРНА РОБОТА №5

Побудова та програмування примітивно-рекурсивного опису функцій, що реалізують найпростіші арифметичні операції

Метою роботи є розробка процедур, що реалізують оператори суперпозиції та примітивної рекурсії, а також розробка програми, що на підставі цих процедур обчислювала б значення функцій, що утворюються із найпростіших (базисних) функцій за допомогою зазначених операторів.

На підставі цих процедур повинна бути складена програма, що обчислює значення заданих примітивно-рекурсивних функцій. У якості заданих пропонуються функції, що реалізують операції додавання, усіченого вирахування, мінімуму, максимуму, модуля вирахування, множення, факторіала числа, зведення в ступінь, операції ділення (цілочислові залишок і частка).

У процесі виконання роботи формуються наступні вміння:

- програмування складних функцій;
- програмування примітивно-рекурсивних описів функцій;
- реалізація рекурсивних викликів функцій і процедур.

Для виконання роботи необхідне вміння використовувати оператори процедур і функцій мовою Object Pascal у середовищі Delphi.

Завдання

Значення ***x*** **= ((номер комп'ютера) mod 7)+3**. Значення ***y*** вибирати невеликим, особливо для "старших" операцій (пункти 8-10) через можливе переповнення стека в ході відкладених викликів рекурсій.

1. Скласти програму, що реалізує функцію додавання $f_+(x,y) = x+y$, використовуючи її примітивно-рекурсивний опис $f_+(x,y) = R_1(I_1^1(x), h(x,y,z))$, де $h(x,y,z) = z+1$ – функція проходження. Для цього необхідно скласти допоміжну процедуру-функцію для обчислення функції $h(x,y,z)$ і використовувати її при рекурсивному обчисленні функції f_+.

2. Скласти програму, що реалізує функцію усіченого вирахування $f_-(x,y) = x-y$, використовуючи її примітивно-рекурсивний опис $f_-(x,y) = R_1(I_1^1(x), h(x,y,z))$, де $h(x,y,z) = I_3^3(x,y,f_1(z)))$, а $f_1(x) = R_0(0, I_2^2(x,y))$ – функція передування $h(z) = z-1$.

3. Скласти програму, що реалізує функцію мінімуму $f_{\min}(x,y) = f_-(x, f_-(x,y))$ з використанням функції усіченого вирахування.

4. Скласти програму реалізації функції максимуму $f_{\max}(x,y) = f_+(y, f_-(x,y))$ з використанням функцій додавання і усіченого вирахування.

5. Скласти програму, що реалізує функцію модуля вирахування $f_{\mathrm{mod}}(x,y) = |x-y| = f_-(x,y) + f_-(y,x)$ з використанням функції усіченого вирахування.

6. Скласти програму, що реалізує функцію залишку від ділення y на x $f_{ost}(x,y) = R_1(0, h(x,y,z))$ де $h(x,y,z) = (z+1)sg(|x-(z+1)|)$, а допоміжна функція знака (сиґнум) $sg(x) = R_0(0, I_2^2(x,1))$.

7. Скласти програму, що реалізує функцію частки від ділення y на x $f_{chasn}(x,y) = R_1(0, h(x,y,z))$ де $h(x,y,z) = z + notsg(|x-(f_{ost}(x,y)+1)|)$, а допоміжна функція заперечення знака (антисиґнум) $notsg(x) = R_0(I_2^2(x,1),0)$.

8. Скласти програму, що реалізує функцію множення $f_{\times}(x,y) = xy$, використовуючи її примітивно-рекурсивний опис $f_{\times}(x,y) = R_1(0, h(x,y,z))$, де $h(x,y,z) = f_{+}(x,z)$. Для цього необхідно, використовуючи попередню програму додавання, скласти допоміжну процедуру-функцію для обчислення функції $h(x,y,z)$ і використовувати її при рекурсивному обчисленні функції $f_{\times}$.

9. Скласти програму, що реалізує функцію факторіала числа $f_{fact}(x) = x!$, використовуючи її примітивно-рекурсивний опис $f_{fact}(x) = R_0(1, h(x,y))$, де $h(x,y) = f_{\times}(x, y-1)$. Для цього необхідно, використовуючи попередню програму множення, скласти допоміжну процедуру-функцію для обчислення функції $h(x,y)$ та використовувати її при рекурсивному обчисленні функції f_{fact}.

10. Скласти програму, що реалізує функцію зведення в ступінь $f_{\exp}(x,y) = x^y$, використовуючи її примітивно-рекурсивний опис $f_{\exp}(x,y) = R_1(1, h(x,y,z))$, де $h(x,y,z) = f_{\times}(x,z)$. Для цього необхідно, використовуючи попередню програму множення, скласти допоміжну процедуру-функцію для обчислення функції $h(x,y,z)$ та використовувати її при рекурсивному обчисленні функції $f_{\exp}$.

Результатом виконання лабораторної роботи є текстовий документ, що містить тексти програм і скріншоти роботи програми при обчисленні функцій по пунктах завдання 1-10. Для кожної функції у звіт помістити схему примітивної рекурсії.

При захисті лабораторної роботи необхідно відповісти на наступні питання.

1. Які функції називаються найпростішими (базисними)?
2. Який оператор називається оператором суперпозиції?
3. Як обчислює функцію оператор примітивної рекурсії?
4. Запишіть схему примітивної рекурсії для обчислення n=5
5. Дайте визначення примітивно-рекурсивної функції.
6. Доведіть примітивну рекурсивність додавання.
7. Доведіть примітивну рекурсивність усіченого вирахування.
8. Доведіть примітивну рекурсивність залишку від ділення.
9. Доведіть примітивну рекурсивність частки від ділення.
10. Доведіть примітивну рекурсивність множення.
11. Що таке примітивно-рекурсивний предикат?
12. Що є характеристичною функцією предиката "x<y"?

ЛАБОРАТОРНА РОБОТА №6

Побудова та програмування примітивно-рекурсивного опису примітивно-рекурсивних операторів

Метою роботи є розробка процедур, що реалізують примітивно-рекурсивні описи операторів підсумовування, перемножування та спільної рекурсії.

Для виконання роботи необхідне вміння використовувати оператори процедур і функцій мовою Паскаль.

Відомості з теорії

Нехай $f(x_1,...,x_n,y)$ – функція від $(n+1)$-ої змінної. Добре відомі операції підсумовування $\sum_{y<z} f(x_1,...x_n,y)$ і перемножування $\prod_{y<z} f(x_1,...x_n,y)$ по змінній y з межею z – це оператори, які з функції $f(x_1,...,x_n,y)$ породжують нові функції $q(x_1,...,x_n,z)=\sum_{y<z} f(x_1,...,x_n,y)$ і $p(x_1,...,x_n,z)=\prod_{y<z} f(x_1,...,x_n,y)$. Покажемо їхню примітивну рекурсивність.

$q(x_1,...,x_n,0)=0$ (за визначенням);

$q(x_1,....x_n,y+1)=q(x_1,...,x_n,y)+f(x_1,...,x_n,y+1)$;

$p(x_1,...,x_n,0)=1$; (за визначенням);

$p(x_1,....x_n,y+1)=p(x_1,...,x_n,y)\cdot f(x_1,...,x_n,y+1)$.

Завдання

1. Скласти програму, що реалізує примітивно-рекурсивний опис оператора підсумовування $f_{summ}(x,y)=\sum_y f1(x,y)$ за схемою $f_{summ}(x,y)=R_1(0,h(x,y,z))$, де $h(x,y,z)=f_+(z,f1(x,y))$. Використовуйте програму додавання з попередньої лабораторної роботи (разом з допоміжними функціями) при створенні процедури-функції $h(x,y,z)$ та опис заданої по варіанту функції $f1(x,y)$.

2. Скласти програму, що реалізує примітивно-рекурсивний опис оператора підсумовування $f_{summ3}(x_1,x_2,y)=\sum_y f3(x_1,x_2,y)$ за наступною схемою $f_{summ3}(x_1,x_2,y)=R_1(0,h(x_1,x_2,y,z))$, де $h(x_1,x_2,y,z)=f_+(z,f3(x_1,x_2,y))$. Використовуйте програму додавання з попередньої лабораторної роботи (разом з допоміжними функціями) при створенні процедури-функції $h(x_1,x_2,y,z)$ та опис заданої по варіанту функції $f3(x_1,x_2,y)$.

3. Скласти програму, що реалізує примітивно-рекурсивний опис оператора перемножування $f_{peremn}(x,y)=\prod_y f2(x,y)$ за схемою $f_{peremn}(x,y)=R_1(1,h(x,y,z))$, де $h(x,y,z)=f_\times(z,f2(x,y))$. Використовуйте програму множення з попередньої

лабораторної роботи (разом з допоміжними функціями) при створенні процедури-функції $h(x,y,z)$ та опис заданої по варіанту функції $f2(x,y)$.

4. Скласти програму, що реалізує примітивно-рекурсивний опис оператора перемножування $f_{peremn3}(x_1,x_2,y)=\prod_y f4(x_1,x_2,y)$ за наступною схемою $f_{peremn3}(x_1,x_2,y)=R_1(1,h(x_1,x_2,y,z))$, де $h(x_1,x_2,y,z)=f_{\times}(z,f4(x_1,x_2,y))$. Використовуйте програму множення з попередньої лабораторної роботи (разом з допоміжними функціями) при створенні процедури-функції $h(x_1,x_2,y,z)$ та опис заданої по варіанту функції $f4(x_1,x_2,y)$.

Варіанти

№ комп.	$f1(x,y)$	$f2(x,y)$	$f3(x_1,x_2,y)$	$f4(x_1,x_2,y)$
1	xy	$(x+1)y$	$(2x_1+x_2)^{y+1}$	$(2x_1x_2+y)3$
2	$x+y$	x^y	$(x_1+2x_2)^{y+1}$	$(x_1+x_2+1)y$
3	$2x+y$	$(x+1)y$	x_1+x_2+2y	$(x_1+x_2)^y$
4	$x+2y$	$(2x)^{y+1}$	$(x_1+x_2)y$	$(x_1x_2)^y$
5	x^y	$x+y$	$2x_1+x_2+y$	$(2x_1+3x_2)^{y+1}$
6	y^x	$x+y$	x_1x_2+y	$(x_1+x_2+1)y$
7	$2xy$	$(x+2y)2$	$(2x_1x_2+y)3$	$(2x_1+x_2)^{y+1}$
8	$(2x+1)y$	xy	$(x_1x_2+1)y$	x_1x_2y
9	$(x+1)y$	$x+y$	$(x_1+x_2+1)y$	x_1+x_2+2y
10	$(x+2y)2$	$x+2y$	$(2x_1x_2)^{y+1}$	$(x_1+x_2)y$
11	$(2x+y)3$	xy	$(x_1x_2)^y$	$2x_1+x_2+y$
12	$(2x)^{y+1}$	$x(y+3)$	$(x_1+2x_2)^y$	x_1x_2+y

Результатом виконання лабораторної роботи є налагоджені програми і текстовий документ, що містить тексти програм і скріншоти роботи програми.

При захисті лабораторної роботи необхідно відповісти на наступні питання.

1. Що таке примітивно-рекурсивний оператор?
2. Який оператор називається оператором умовного переходу?
3. Напишіть схему спільної рекурсії для n=5, k=3.
4. Доведіть примітивну рекурсивність обмеженого оператора мінімізації.
5. Доведіть примітивну рекурсивність оператора умовного переходу.
6. Запишіть схему примітивної рекурсії оператора підсумовування для нижньої межі більшої ніж 0.
7. Запишіть схему примітивної рекурсії оператора перемножування для нижньої межі більшої ніж 0.
8. Дайте визначення частково-рекурсивної функції, загальнорекурсивної функції.
9. Сформулюйте тезу Черча.
10. Як виконується арифметизація конфігурацій машини Т'юринга? Запишіть четвірку для конфігурації $11011q_901101$.

ЛАБОРАТОРНА РОБОТА №7

Програмування функцій Акермана і обчислення їхніх значень

Метою роботи є розробка програм, що здійснює обчислення значень функцій Акермана.

У процесі виконання роботи формуються наступні вміння:

- програмування складних функцій;
- програмування рекурсивних описів функцій;
- реалізація рекурсивних викликів функцій і процедур.

Для виконання роботи необхідне вміння використовувати оператори процедур і функцій мовою Паскаль.

Завдання

1. Скласти програму, що здійснює обчислення функції $B(x, y)$ використовуючи її рекурсивний опис:

$$B(0, y) = 2 + y;$$
$$B(x + 1, 0) = sg(x);$$
$$B(x + 1, y + 1) = B(x, B(x + 1, y)).$$

2. Скласти програму, що обчислює функцію $A(x) = B(x, x)$, використовуючи процедури, складені при виконанні п. 1

Результатом виконання лабораторної роботи є програма, а також текстовий документ, що містить тексти основної програми і допоміжних процедур. Результати обчислень представити у вигляді скріншотів роботи програми.

При захисті лабораторної роботи необхідно відповісти на наступні питання.

1. Яка функція називається В-мажоровуваною?
2. У чому складається властивість функції $B(x, y)$ неубування по другому аргументу?
3. У чому складається властивість функції $B(x, y)$ зростання по другому аргументу?
4. У чому складається властивість функції $B(x, y)$ зростання по першому аргументу?
5. Сформулюйте теорему про ріст функції Акермана. У чому суть доказу цієї теореми?

СПИСОК ЛІТЕРАТУРИ

1. Клини С.К. Введение в математику. М.: Ин. л-ра, 1957.
2. Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. М.: Наука, 1984. – 224 с.
3. Мальцев А.И. Алгоритмы и рекурсивные функции. М.: 1986.
4. Трахтенброт Б.А. Алгоритмы и вычислительные автоматы. М.: Советское радио, 1974.
5. Роджерс Х. Теория рекурсивных функций и эффективная вычислимость. М.: Мир, 1972.
6. Минский М. Вычисления и автоматы. М.: Мир, 1971.
7. http://www.ict.edu.ru/ft/004979/Posob3.pdf (Учебное пособие— Бильгаева Н.Ц. Теория алгоритмов, формальных языков, грамматик и автоматов, 2000)
8. http://www.klgtu.ru/students/literature/teoralgor_ta.pdf (Учебное пособие— Пономарев В.Ф. Основы теории алгоритмов, 2005)

Укладачі:
Трусов Валерій Олександрович
Зацепін Євген Павлович

ТЕОРІЯ АЛГОРИТМІВ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ І ЗАВДАННЯ

до виконання лабораторних робіт з дисципліни
для студентів
напряму підготовки 6.050101 Комп'ютерні науки

Друкується в редакційній обробці укладачів.

Підписано до друку 25.06.12. Формат 30х42/4.
Папір офсет. Ризографія. Ум. друк. арк. 1,3.
Обл.-вид. арк. 1,3. Тираж 40 пр. Зам. № .

Державний вищий навчальний заклад
"Національний гірничий університет"
49005, м. Дніпропетровськ, просп. К. Маркса, 19