МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Кафедра обчислювальної техніки

МЕТОДИЧНІ ВКАЗІВКИ

до виконання курсової роботи
з дисципліни
"Системне програмування"

Розробник: доцент, канд. техн. наук, доцент Павлов В.Г.
(посада, вчена ступінь та звання П.І.Б.)

Затверджено на засіданні кафедри
Протокол № 1 від 30 серпня 2021 р.

Завідувач кафедри ОТ
_____ Стіренко С.Г.
(підпис) (прізвище, ініціали)

Київ – 2022

Мета роботи: Одержання навичок складання системних програм у відповідності до вимог завдання або технічного завдання і підготовки комплекту документів на програми для її замовника або користувача. Вивчення основних вимог до базових документів системних програм та одержання навичок оформлення таких документів на прикладах реалізації системних програм.

Вимоги до оформлення:

Пояснювальна записка (ПЗ) до курсової роботи виконується у форматі MsWord та у відповідності до вимог Єдиної Системи Програмної Документації (ЄСПД) і починається двома стандартними аркушами – титульний аркуш та аркуш завдання, якій наведений у Додатку А. Далі структура ПЗ така:

1. Технічне завдання на КР.
2. Зміст, з усіма розділами, документами тощо. Заголовки розділів супроводжують номери сторінок їх початку.
3. Вступ, в якому наводиться зміст завдання, аргументується корисність та актуальність теми роботи в професійній діяльності.
4. Огляд методів побудови системних програм даного типу з аналізом їх переваг та недоліків. Обґрунтування вибору даного методу. Загальна блок-схема компілятору.
5. Опис роботи вхідної програми за блок-схемою у відповідності зі стандартами ГОСТ 19.402-78 ЕСПД, ГОСТ 19.701-90 ЕСПД. Опис програми повинен включати вичерпну інформацію про формати вхідних та вихідних даних. Опис алгоритму програми повинен пояснювати ключові елементи алгоритму та виклики стандартних та спеціальних функцій. Стандартні і спеціальні програми загального використання також повинні бути документовані у повному або розширеному комплекті документації за стандартом ГОСТ 19.402-78 ЕСПД. Блок-схема і текст програми оформлюються за стандартами ЄСПД і наводяться у тексті або у Додатках до пояснювальної записки.
6. Лістинги програми компілятору в цілому або окремих її модулів, що розроблялися під час виконання курсової роботи.
7. Тестування роботи програми виконується за допомогою контрольних прикладів. Контрольні приклади повинні бути побудовані таким чином, щоб максимально перевірити та продемонструвати усі функціональні можливості розробленого компілятора в тому числі і у обробці помилок. Кожний валідний контрольний приклад супроводжується зробленими розрахунками, порівняння з якими підтверджує працездатність програми. Кількість контрольних прикладів визначається кількістю можливих ситуацій зі вхідними даними, але не може бути менше ніж по п'ять (валідних та помилкових). Порівнянні виконується як для програми на вхідної мови, так і для коду, згенерованого на мові Assembler x86. ***Наявність інформативних скріншотів обов'язкова***.
8. Висновки, в яких обґрунтовується оцінка стану розробленої програми, її відповідність вимогам завдання. Аналізуються недоліки і помилки в програмі, а також шляхи їх виправлення.
9. Список використаної літератури.
10. Додатки (за потребою)

Текст рекомендовано готувати в текстовому процесорі з використанням шрифту Times New Roman (14pt) з полуторним інтервалом між рядками.

Щоб раціонально створювати програми і програмні комплекси, важливо використовувати принципи **модульного програмування**, тобто оформлення основи програми у вигляді комплексів процедур, функцій або інших підпрограм, які утворюють **модулі широкого використання** або **бібліотечні модулі мов програмування**. Базовий текст прикладних або системних програм також будується у вигляді окремого **управляючого модуля програми** або **модуля тестування**, які є головними модулями програми і включають набір контрольних прикладів або закінчену програму автономного тестування.

Склад програмного пакету, який супроводжує файл ПЗ до КР:

1. Програма на вхідної мові, яка вирішує поставлене завдання (файл "cpp" або "py".
2. Програма компілятору у вигляди виконуваного файлу або bat-файлу та файлу «jar» для Java.
3. Вихідний файл у форматі MASM32 або асемблерної вставки.
4. Якщо вихідний файл у форматі асемблерної вставки, то файл з вже вбудованою цією вставкою.

Усі вказані файли повинні знаходиться виключно у кореневому каталозі, усі інші файли розміщуються у створеної внутрішньої папці.

Шаблон для назв файлів та усіх папок, а також для надсилаємого архіву:

КР-<номер варіанту>-<мова реалізації компілятору>-<номер групи>-<прізвище>

Наприклад: **КР-22-Java-ІВ-92-Petrov** (усі літери латинські)

ДОДАТОК А. Лист технічного завдання (приклад)

Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"

Факультет (інститут) інформатики та обчислювальної техніки
(повна назва)

Кафедра обчислювальної техніки
(повна назва)

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 123 «Комп'ютерна інженерія»
(шифр і назва)

З А В Д А Н Н Я

НА КУРСОВУ РОБОТУ СТУДЕНТУ

Петренка Сергія Петровича
(прізвище, ім'я, по батькові)

1. Тема роботи «Розробка компілятора підмножини команд мови С (або Python), що містять арифметичні дії і дужкові форми»
керівник роботи Павлов В.Г. к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи грудня 2021 р.

3. Вхідні дані до роботи (приклад)
- вхідна мова: С / Python;
- цільова мова: Assembler x86;
- типи конструкцій: арифметичні дії, розгалуження, цикли, функції.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- розробити лексичний аналізатор мови С / Python;
- розробити синтаксичний аналізатор мови С / Python;
- розробити генератор коду мови Assembler x86.

5. Перелік графічного матеріалу;
- алгоритм лексичного аналізу;
- алгоритм синтаксичного аналізу;
- дерева розбору.

6. Дата видачі завдання 15 жовтня 2021 р

Завдання:

1. Варіант завдання та вхідна мова програмування визначається номером у списку групи. Кожному варіанту відповідає номер алгоритму. Знайти необхідний алгоритм, реалізувати його на мові програмування відповідного варіанту та отримати множину результатів контрольних прикладів, як валідних, так й тих, що містять помилку у вхідному коді.
2. Програма реалізації алгоритму на вхідної мові повинна містити коментарі, що описують роботу цього алгоритму.
3. Для перевірки кожного валідного результату навести відповідні математичні обрахунки з проміжними результатами.
4. Розробити програму компілятора, яка здійснює генерацію асемблерного коду MASM32, відповідно до вхідної програми.
5. Реалізувати обробку коментарів у компіляторі та вивід результату у віконному інтерфейсі.
6. Здійснити перевірку тотожності результатів виконання асемблерних програм контрольнім результатам, отриманих у пункті 1.
7. За підсумками роботи зробити висновки.

Варіант	Номер алгоритму	Вхідна мова
1	4	C
2	6	Python
3	1	C
4	3	Python
5	10	C
6	5	Python
7	2	C
8	8	Python
9	7	C
10	9	Python
11	11	C
12	12	Python
13	13	C
14	14	Python
15	15	C

Варіант	Номер алгоритму	Вхідна мова
16	2	Python
17	6	C
18	4	Python
19	5	C
20	1	Python
21	8	C
22	7	Python
23	12	C
24	10	Python
25	9	C
26	11	Python
27	3	C
28	13	Python
29	14	C
30	15	Python

Алгоритми для реалізації:

1. **Числа Фібоначчі. Рекурсивно**. Реалізуйте функцію знаходження n-го числа послідовності Фібоначчі за допомогою рекурсивної функції. Вхідними даними є номер необхідного числа послідовності; вихідним є ціле невід'ємне число, що належить до послідовності.
2. **Факторіал. Рекурсивно**. Реалізуйте функцію знаходження факторіала числа рекурсивним чином. Вхідними даними є невід'ємне ціле число, факторіал якого треба знайти; вихідним є ціле невід'ємне число.
3. **Зміна регістру символу**. На вхід приймається символ або ASCII-код символу у мові Python у *нижньому* регістрі, на виході отримуємо ASCII код цього ж символу у *верхньому* регістрі.
4. На вхід подати два цілих числа. Знайти **максимальне** з них за допомогою тернарного оператора. Знайти **суму** усіх **цілих дільників** цього числа.
5. **Простий калькулятор**. Вхідними параметрами є 2 операнди будь-якого типу та номер операції. Відповідність *номер операції-операція* відобразити у коментарях. Операції, які підтримує калькулятор: усі бінарні та унарні операції, що розглядалися у курсі лабораторних робіт.
6. **Прості числа**. Реалізувати алгоритм, що виводить суму усіх простих чисел у певних межах, які задаються користувачем. Межі є невід'ємними числами будь-якого чисельного типу.
7. Дано натуральне число N. Порахувати **суму перших n доданків** за принципом - (2*1- 2*3- 4*3- 4*5- 6*5- 6*7-…).
   Порахувати **добуток перших N співмножників** за принципом (1/1)* (4/2)* (8/3)* (16/4)...
   Операцію ділення виконувати тільки якщо її результат є цілим числом.
   Знайти *максимальне* число серед двох отриманих результатів.
8. **Алгоритм Евкліда**. Реалізувати алгоритм Евкліда для знаходження найбільшого спільного дільника та найменшого спільного кратного. На вхід подати невід'ємне ціле число.
9. **Арифметична прогресія.** Знайти n-й член арифметичної прогресії. Знайти суму перших n членів прогресії. Перший елемент, n-й елемент та різниця задаються у якості параметрів і є цілими невід'ємними числами.
10. **Геометрична прогресія.** Знайти n-й член геометричної прогресії. Знайти суму перших n членів прогресії. Перший елемент, n-й елемент та різниця задаються у якості параметрів і є цілими невід'ємними числами.
11. **Числа Фібоначчі. Ітераційно**. Реалізуйте функцію знаходження n-го числа послідовності Фібоначчі за допомогою ітерації. Вхідними даними є номер необхідного числа послідовності; вихідним є ціле невід'ємне число, що належить до послідовності.
12. **Факторіал. Ітераційно.** Реалізуйте функцію знаходження факторіала числа використовуючи цикл. Вхідними даними є невід'ємне ціле число, факторіал якого треба знайти; вихідним є ціле невід'ємне число.

13. **Зміна регістру символу**. На вхід приймається символ або ASCII-код символу у мові Python у *верхньому* регістрі, на виході отримуємо ASCII код цього ж символу у *нижньому* регістрі.
14. На вхід подати два цілих числа. Знайти **мінімальне** з них за допомогою тернарного оператора. Знайти **суму** усіх **цілих дільників** цього числа.
15. Невід'ємне ціле число n задається користувачем. Знайти суму перших n членів послідовності, що задається формулою $n * (-1)^n$.

Література:

Основна:

1. Альфред В. Ахо, Моника С. Лам и др. Компиляторы. Принципы, технологии, инструменты. Пер. с англ. – М, : Вильямс, 2016, - 1184 с.
2. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Пер. с англ. В 2-х т. – М, : Вильямс, 2012, - 464 с.
3. Вирт Никлаус. Построение компиляторов. Пер. с англ. – М, : ДМК Пресс, 2010, - 192 стр.

Допоміжна:

1. Yale N. Patt, Sanjay J. Patel. Introduction To Computing Systems: From Bits And Gates To C And Beyond. Singapore, 2005. URL: https://www.academia.edu/34254842/INTRODUCTION_TO_COMPUTING_SYSTEMS_FROM_BITS_AND_GATES_TO_C, International_Edition_2005
2. Abdulaziz Ghuloum. An Incremental Approach to Compiler Construction, Conference: Scheme and Functional Programming Workshop , 2006. URL: http://scheme2006.cs.uchicago.edu/11-ghuloum.pdf.
3. ДСТУ ГОСТ 2.104:2006 ЄСКД. Основні написи.
4. ДСТУ ГОСТ ИСО 8790:2003. Системи оброблення інформації. Символи й умовні познаки для схем конфігурації обчислювальної системи
5. ДСТУ ГОСТ 2.702:2013 ЄСКД. Правила виконання електричних схем.
6. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.
7. ДСТУ ISO 5807:2016 (ГОСТ 19.701-90). Обробляння інформації. Символи та угоди щодо документації стосовно даних, програм та системних блок-схем, схем мережевих програм та схем системних ресурсів