Отчёт по лабораторной работе

№ 6

Модульное программирование

Задание. Для задачи из п.5 лабораторной работы 5:

1. Дополнительно предусмотреть возможность ввода с клавиатуры нескольких символов последовательно

**1. Постановка задачи**

Разработать консольную программу, которая позволяет пользователю вводить несколько символов подряд, выполняя заданные действия для каждого символа в зависимости от выбранного варианта использования. Программа должна корректно обрабатывать ошибки ввода и завершать работу по запросу.

**2. Входные и выходные данные**

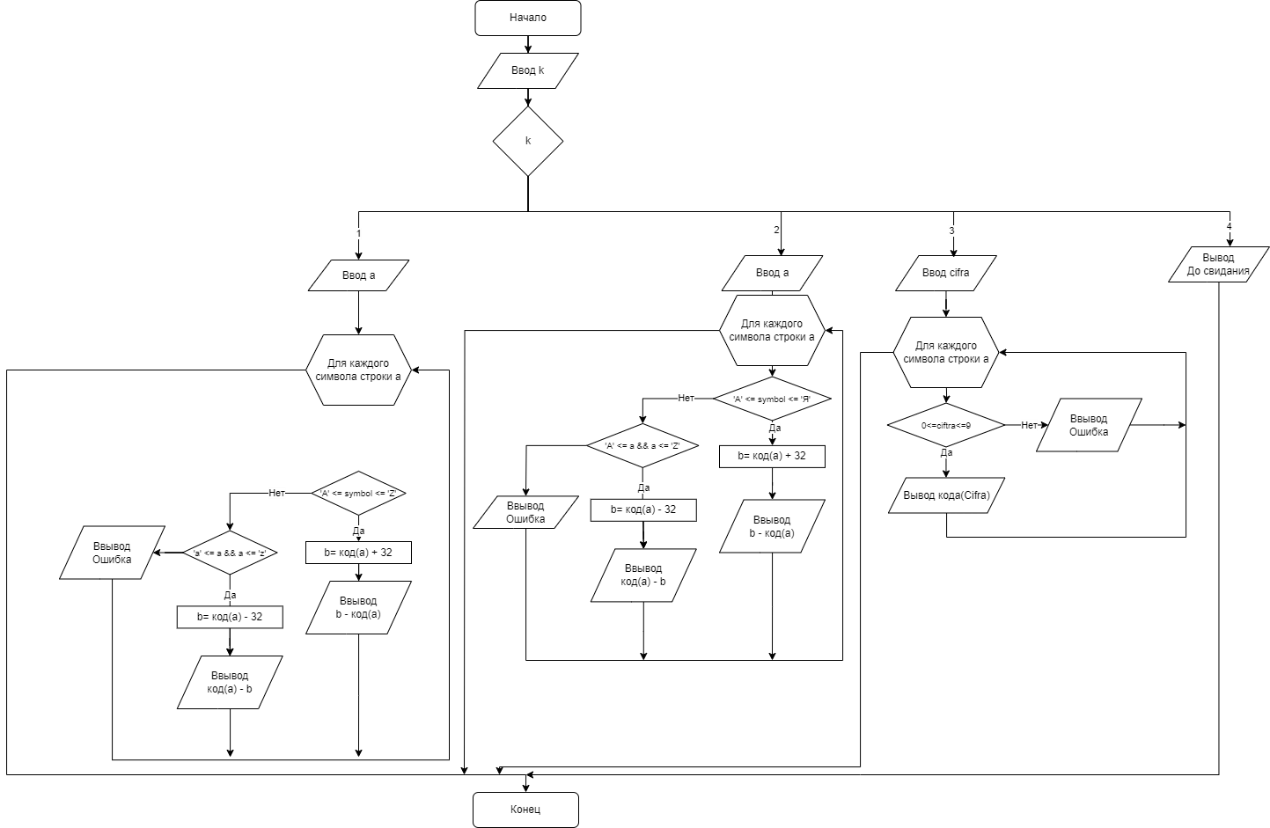
**Входные данные:**

1. **Строка символов, введенная пользователем.**  
   Пользователь вводит несколько символов подряд.
2. **Номер операции (1, 2, 3 или 4).**  
   Пользователь выбирает вариант использования для обработки символов:
   * **1**: Проверка латинских букв (разница ASCII-кодов).
   * **2**: Проверка русских букв (разница Windows-1251).
   * **3**: Проверка цифр (вывод кода).
   * **4**: Завершение программы.

**Выходные данные:**

1. Для каждого символа:
   * Результат выполнения операции (например, разница кодов или код символа).
   * Сообщение об ошибке, если символ не подходит под выбранную операцию.
2. Сообщение о завершении работы программы.

**БЛОК-СХЕМ:**

****

**6. Декомпозиция**

1. Модуль нахождения разницы между латинскими символами:

**Входные данные** строка а

**Выходные данные** число являющиеся разницой между двумя символами

**ПСЕВДОКОД:**

НАЧАЛО

ВВОД строку а

ДЛЯ КАЖДОГО символа СТРОКИ а

НАЧАЛО ЦИКЛА

ЕСЛИ А<= символа <=Z ИЛИ а<= символ <= Z

ВЫВОД РАЗНИЦА КОДОВ СИМВОЛОВ

ИНАЧЕ

ВЫВОД ОШИБКА

КОНЕЦ ЦИКЛА

КОНЕЦ

1. Модуль нахождения разницы между русскими символами:

**Входные данные** строка а

**Выходные данные** число являющиеся разницой между двумя символами

**Псевдокод:**

НАЧАЛО

ВВОД строку а

ДЛЯ КАЖДОГО символа СТРОКИ а

НАЧАЛО ЦИКЛА

ЕСЛИ А<= символа <=Я ИЛИ а<= символ <= я

ВЫВОД РАЗНИЦА КОДОВ СИМВОЛОВ

ИНАЧЕ

ВЫВОД ОШИБКА

КОНЕЦ ЦИКЛА

КОНЕЦ

1. Модуль вывода символа по коду введённому с клавиатуры

**Входные данные** строка а

**Выходные данные** код введённого символа

**Псевдокод:**

НАЧАЛО

ВВОД строку а

ДЛЯ КАЖДОГО символа СТРОКИ а

НАЧАЛО ЦИКЛА

ЕСЛИ 0<= символа <=9

ВЫВОД код введённой цифры

ИНАЧЕ

ВЫВОД ОШИБКА

КОНЕЦ ЦИКЛА

КОНЕЦ

Пункт 7:

Program\_lab06

Начало программы

**DO**

Выбор операции

Для каждого символа в строке:

ВЫБОР 1:

Проверить, является ли символ латинской буквой

Если да, вычислить разницу ASCII-кодов

Иначе, вывести сообщение об ошибке

ВЫБОР 2:

Проверить, является ли символ русской буквой

Если да, вычислить разницу кодов Windows-1251

Иначе, вывести сообщение об ошибке

ВЫБОР 3:

Проверить, является ли символ цифрой

Если да, вывести код символа

Иначе, вывести сообщение об ошибке

ВЫБОР 4:

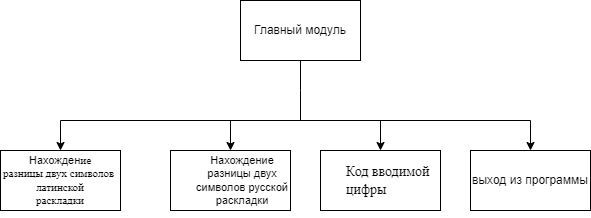
Выход из программы

**While(k!=4)**

Конец цикла

Конец программы

Пункт 8:



Псевдокод:

НАЧАЛО

ДЕЛАЙ

ВВОД k

1 ВЫБОР

Нахождение разницы двух символов латинской раскладки

2 ВЫБОР

Нахождение разницы двух символов русской раскладки

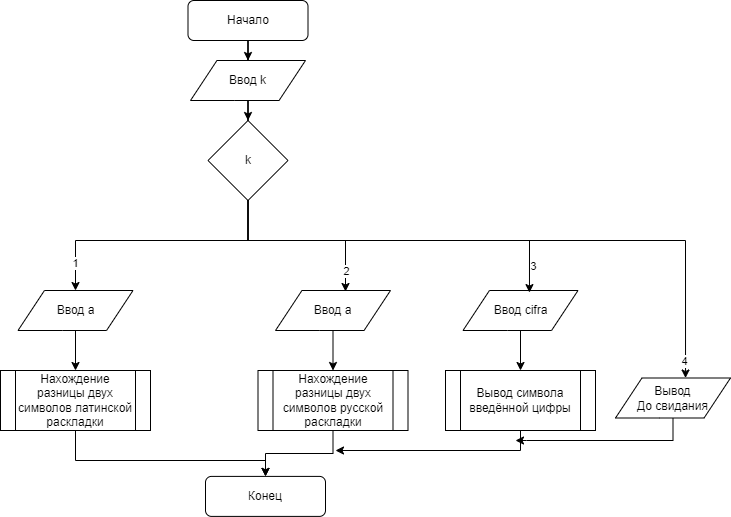
3 ВЫБОР

Код вводимой цифры

4 ВЫБОР

ПОКА k != 4

КОНЕЦ

****

Ответы на вопросы

1. Парадигма программирования – это совокупность идей и понятий, определяющих стиль написания компьютерных программ (подход к программированию). Структурное программирование, Объектно-ориентированное программирование, Функциональное программирование, Логическое программирование, Визуальное программирование.

2. Методология включает в себя модель вычислений для данного стиля. Методология разработки программного обеспечения – совокупность методов, применяемых на различных стадиях жизненного цикла программного обеспечения.

3. Императивное программирование предполагает, что программа явно описывает алгоритм решения конкретной задачи (действия исполнителя), т.е. описывает как решать поставленную задачу. Декларативное программирование – это предварительная реализация «решателя» для целого класса задач. Тогда для решения конкретной задачи этого класса достаточно декларировать в терминах данного языка только её условие: (исходные данные + необходимый вид результата) «Решатель» сам выполняет процесс получения результата, реализуя известный ему алгоритм решения.

4. Цели структурного программирования: повысить надежность программ (улучшить структуру программы); создание понятной, читаемой программы, которая исполняется, тестируется, конфигурируется, сопровождается и модифицируется без участия автора (создание ПП).

5. следование; ветвление; цикл.

6. нисходящее проектирование (проектирование сверху вниз). модульное (процедурное) программирование. структурное кодирование

7. Модульное программирование – это организация программы как совокупности небольших независимых блоков, называемых модулями. Модуль – функционально законченный фрагмент программы, оформленный в виде отдельного файла с исходным кодом.

8. Функциональная декомпозиция задачи – разбиение большой задачи на ряд более мелких, функционально самостоятельных подзадач – модулей.

9. Модуль – это фрагмент описания процесса, оформленный как самостоятельный программный продукт, пригодный для многократного использования. Модуль, основные характеристики:  один вход и один выход (на вход программный модуль получает набор исходных данных, выполняет их обработку и возвращает набор выходных данных);  функциональная завершенность (модуль выполняет набор определенных операций для реализации каждой отдельной функции, достаточных для завершения начатой обработки данных);  логическая независимость (результат работы данного фрагмента программы не зависит от работы других модулей);  слабые информационные связи с другими программными модулями (обмен информацией между отдельными модулями должен быть минимален);  размер и сложность программного элемента должна быть в разумных рамках.

10. Метод восходящей разработки, метод нисходящей разработки.