Федеральное государственное бюджетное образовательное учреждение высшего

образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа № 2

Подпрограммы

Вариант 20

Выполнил: Назаров Ярослав Сергеевич

студент группы ИВТ-42-23

Проверила: кандидат тех. наук

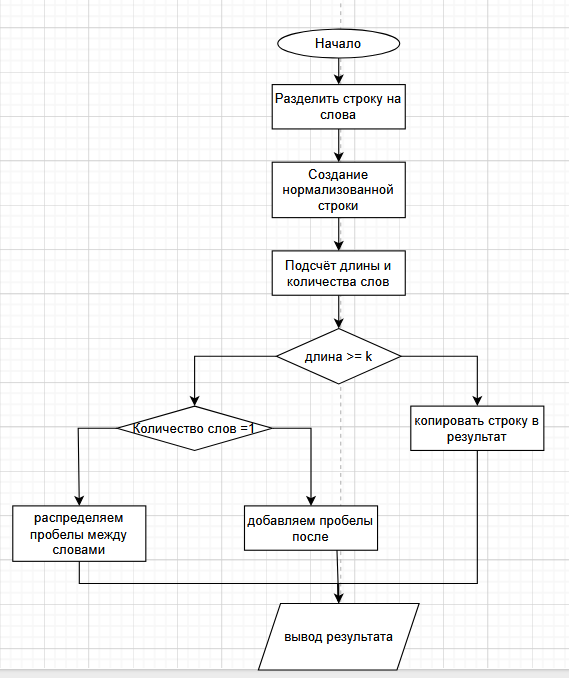
Андреева Антонина Аркадьевна

Чебоксары, 2025

**Цель работы:** изучение подпрограмм на языке Turbo Assembler

**Выполнение работы:**

20. function InsBlanks(S: string; Len: byte): string. Возвращает   
строку длиной Len, в которой равномерно добавлены пробелы.   
Для этого строка циклически просматривается и после каждого   
слова в ней добавляется по одному пробелу до тех пор, пока не   
будет равенства длины строки и Len. Если длина S сразу больше   
или равна Len либо S – пустая строка, то ничего не изменяется.   
Примечание: пустой называется строка, не содержащая ни   
одного символа, байт длины которой равен нулю.



**Программа на высоком языке программирования:**

def ins\_blanks(s, k):

    if not s:

        return s if k <= 0 else ' ' \* k

    # Нормализация строки (удаление лишних пробелов)

    normalized = []

    in\_word = False

    for char in s:

        if char == ' ':

            if in\_word:

                normalized.append(' ')

                in\_word = False

        else:

            normalized.append(char)

            in\_word = True

    # Удаляем последний пробел если есть

    if normalized and normalized[-1] == ' ':

        normalized.pop()

    normalized = ''.join(normalized)

    n = len(normalized)

    if n >= k:

        return normalized

    # Подсчет слов

    words = normalized.split() if normalized else []

    word\_count = len(words)

    if word\_count == 0:

        return ' ' \* k

    if word\_count == 1:

        return words[0] + (' ' \* (k - len(words[0])))

    # Вычисление дополнительных пробелов

    total\_spaces = k - n

    gaps = word\_count - 1

    base\_spaces = total\_spaces // gaps

    extra\_spaces = total\_spaces % gaps

    # Построение результата

    result = []

    for i, word in enumerate(words):

        result.append(word)

        if i < gaps:

            # Базовые пробелы + дополнительный если нужно

            spaces = base\_spaces + 1 if i < extra\_spaces else base\_spaces

            result.append(' ' \* spaces)

    return ''.join(result)

‘

**Стек:**

|  |  |  |
| --- | --- | --- |
| **Смещение** | **Содержимое** | **Описание** |
| [BP+0] | Сохраненное значение BP | Старый указатель базы |
| [BP+2] | Адрес возврата (IP) | Адрес следующей инструкции после call |
| [BP+4] | temp\_buffer (смещение) | Адрес буфера для нормализованной строки |
| [BP+6] | words\_buffer (смещение) | Адрес массива указателей на слова |
| [BP+8] | result\_buffer (смещение) | Адрес буфера для результата |
| [BP+10] | k (значение типа word) | Желаемая длина строки |
| [BP+12] | input\_string (смещение) | Адрес исходной строки |

**Пример выполнения программы на ассемблере:**  
Входная строка: Hello asm

Выходная строка:



**Вывод:** изучил подпрограммы и применил их на практике.