**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение**
**высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА**

# О Т Ч Е Т

**По лабораторной работе** 5

**Вариант 10.**

**Название:** Программирование с использованием разноязыковых модулей.

**Дисциплина:** Машинно-зависимые языки и основы компиляции.

Студент ИУ6-44Б _____ Гуляева В. А.
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Кузнецов Н.О.
(Подпись, дата) (И.О. Фамилия)

Москва, 2020

**Цель работы:** изучение конвенций о способах передачи управления и данных при вызове из программы, написанной на языке высокого уровня, подпрограмм, написанных на ассемблере.

**Задание.**

Разработать программу, состоящую из трех модулей:

– основной модуль на выбранном языке программирования общего назначения должен содержать диалоговый ввод необходимых данных, вызов функции или процедуры на ассемблере и вывод результатов;

– второй модуль - функция или процедура на ассемблере, выполняющая заданную обработку и вызывающая для печати диагностических сообщений процедуру на выбранном языке программирования общего назначения;

– третий модуль - процедура на выбранном языке общего назначения, обязательно получающая некоторые параметры из вызывающего модуля.

Дан текст не более 255 символов. Слова отделены друг от друга пробелами. Текст разбит на строки указанной длины с учетом неделимости слов. Выровнять правую границу текста за счет добавления пробелов между словами.

Первый модуль: основная программа main содержит ввод необходимых данных (текст и длина строк), необходимые преобразования для дальнейшей работы с данными, вызов процедуры ADD1 на ассемблере и вывод результатов (смотри рисунок 1).

Второй модуль: процедура ADD1 разбивает текст на строки заданной длины с учётом неделимости слов, определяет длину наибольшей строки и вызывает процедуру Print (смотри рисунок 3).

Третий модуль: процедура Print выравнивает строки по максимальной длине путём добавления пробелов между словами (смотри рисунок 2).

На рисунках 4 и 5 изображены стек и структурная декомпозиция соответственно.

**Файл ‘Source.cpp’:**

```
#include <iostream>
#include <string>
#include <string.h>

using namespace std;

extern "C" void __cdecl ADD1(char *a, int n);

extern "C" void Print(char *a, int n) //выравниваниепо максимальной строке
{
	char help[125] = "";
	char b[125] = "";
	int j = 0;
	while (a[j] != '\0') {
		for (int i = 0; i < n; i++) {
			help[i] = ' ';
		}
		int s = 0;
		int k = j;
		while (a[k] != '\n') {
			s++;
			k++;
		}
		int flag = 0;
		int prob = n - s;
		int i = 0;
		while (a[j] != '\n') {
			if (flag == 1 || a[j] != ' ') {
				help[i] = a[j];
				i++;
				j++;
			}
			if (a[j] == ' ' && flag == 0) {
				i += prob + 1;
				j++;
				flag = 1;
			}
		}
		help[n] = '\n';
		strcat_s(b, help);
		j++;
	}
	strcpy_s(a, 125, b);
}

int main()
{
	char a[125] = "";
	int n;
	cout << "Input Str1: ";
	gets_s(a);
	cout << "\n";
	cout << "Input len: ";
	cin >> n;
```

```
	cout << "\n";
	int l = strlen(a);
	a[l] = ' '; // добавила в конец пробел
	for (int i = l + 1; i < 125; ++i)
	{
		a[i] = '\0';
	}
	ADD1(a, n);  //разбиение текста на строки
	cout << a;
	system("pause");
	return 0;
}
```

**Файл ‘asm.asm’:**

```
                        .586
                        .model flat
                        .code
                        public _ADD1
                        externdef _Print:near ;подпрограмма на C++
_ADD1 proc
                        push EBP
                        mov  EBP,ESP
                        push ESI
                        push EDI
                        push EAX
                        push EBX
                        push EDX
                        xor  EDX,EDX
                         mov  EDI,[EBP+8]     ;записываю адрес строки
Str1
                          mov  ESI,[EBP+12]   ;записываю n

                        cld
                        mov EDX, ESI ; максимум

                        cycle1:
                              add EDI, ESI; ставлю указатель на n-ый
символ
                              mov AL, 0  ; пока не достигнут конец строки
                              scasb
                             je continue; если конец строки
                             mov EBX, ESI; задаю максимум = n
                             dec EDI    ; возвращаюсь к текущему символу
                              mov AL, ' '
                              cycle2:
                                    scasb      ; проверяю символ на пробел
                                    je next_str ; переход по равно
                                    inc EBX    ; увеличиваю максимум
                              jmp cycle2

                             next_str:
                              cmp EDX, EBX; сравниваю с максимумом
```

```
            jge mx     ; переход по больше или равно
            mov EDX, EBX    ; назначаю новый максимум
            mx:
            dec EDI    ;возвращаюсь к текущему символу
            mov AL, 10
            mov [EDI], AL   ; меняю пробел на символ 10
            inc EDI    ; перехожу к следующему символу
        jmp cycle1

         continue:
         ; преобразования строки для дальнейшей работы
         mov AL, 0
         cyc:
        dec EDI
         cmp [EDI], AL
         je cyc
         mov AL, ' '
         cmp [EDI], AL
         je next
         inc EDI
         next:
         mov AL, 10
         mov [EDI], AL
         inc EDI
         mov AL, 0
         mov [EDI],AL

         mov ESI, [EBP+8] ; записываю адрес строки в ESI
         push EDX          ; записываю максимум в стек
         push ESI          ; записываю адрес строки в стек
         call _Print       ; вызов процедуры
         pop ESI
         pop EDX

         pop  EDX
         pop  EBX
         pop  EAX
         pop  EDI
         pop  ESI
         pop  EBP
         ret
_ADD1 endp
       End
```

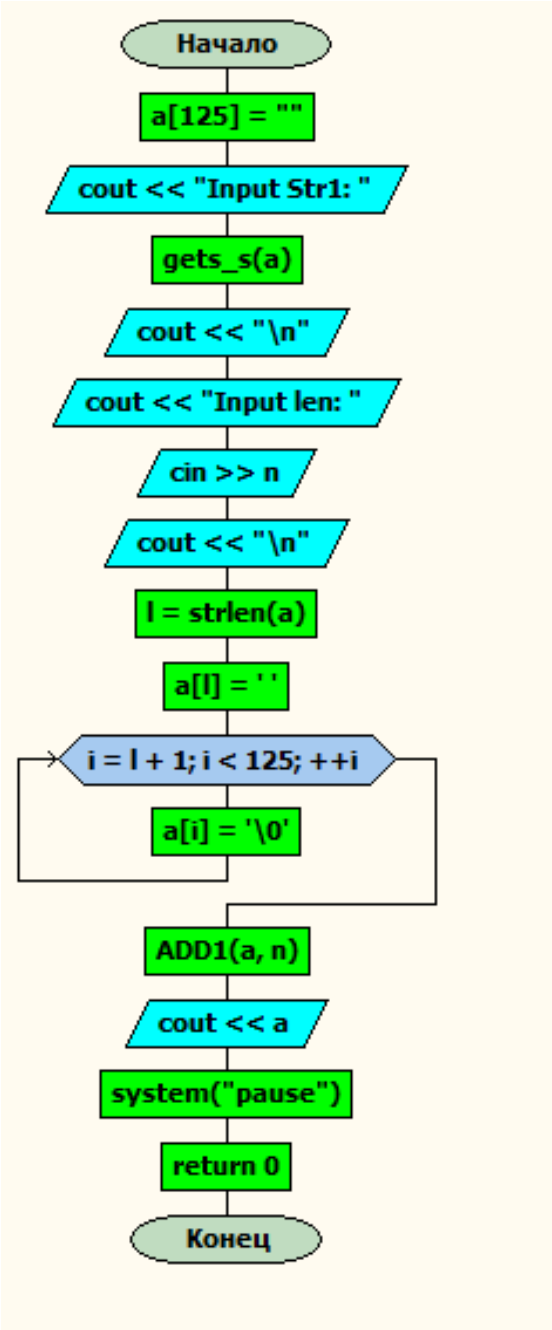


Рисунок 1 – Схема алгоритма первого модуля

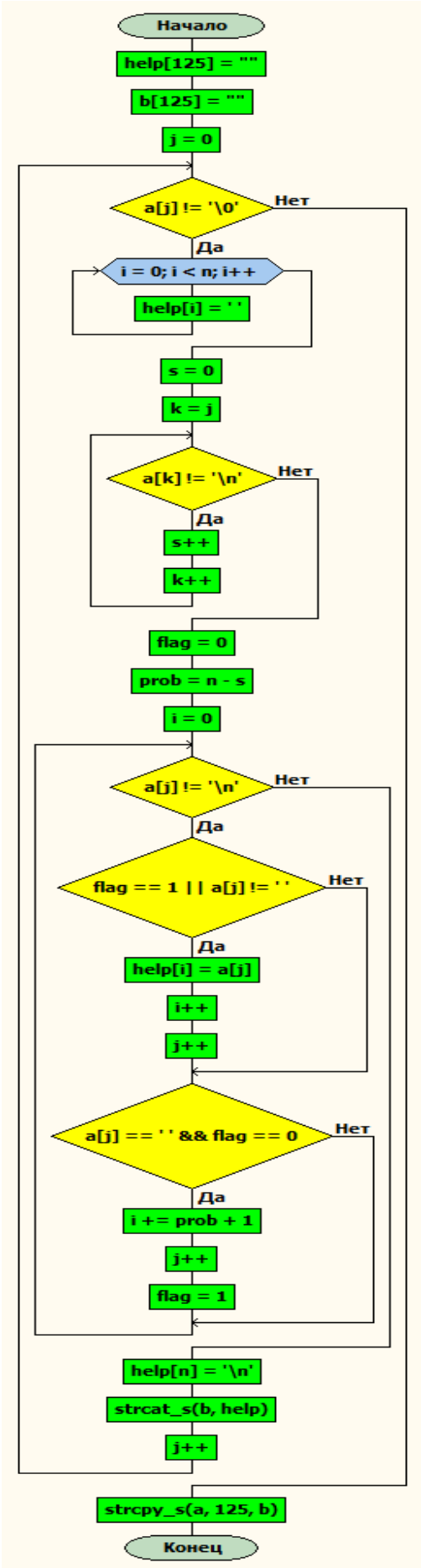


Рисунок 2 – Схема алгоритма подпрограммы на C++

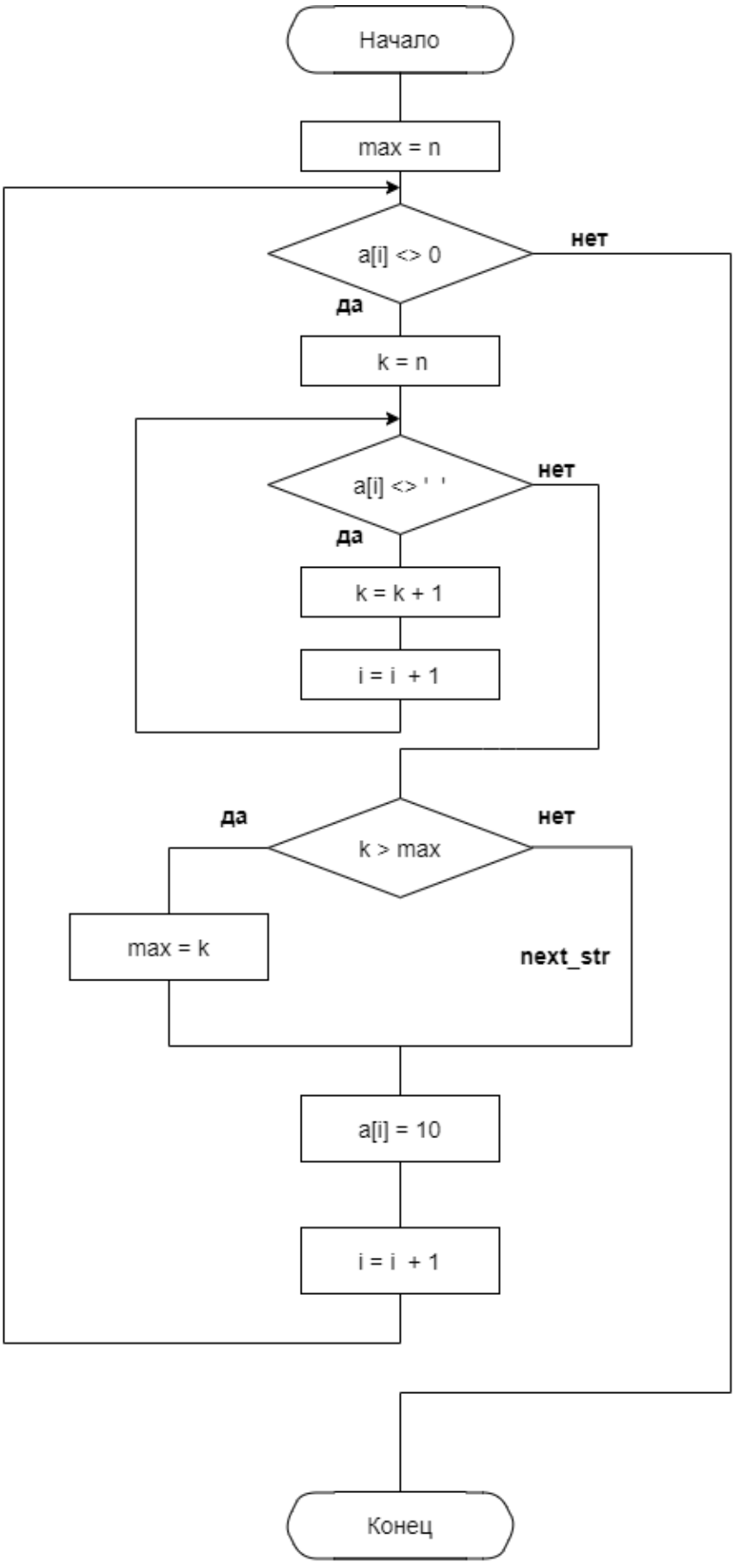


Рисунок 3 – Схема алгоритма подпрограммы на ассемблере

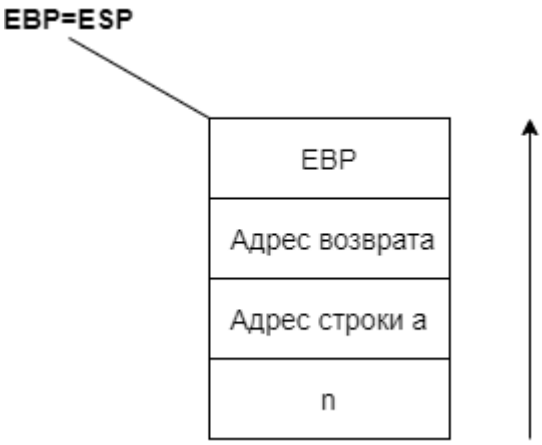


Рисунок 4 – Изображение стека в момент передачи управления подпрограмме на ассемблере

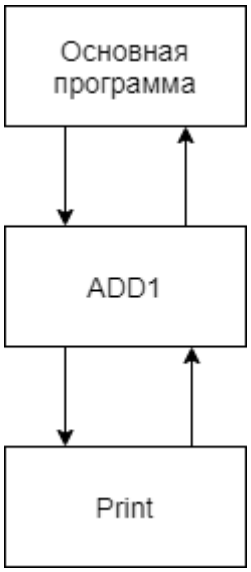


Рисунок 5 – Структурная декомпозиция

## Тестирование.

Таблица 1 – Результаты тестирования.

| Исходные данные | Ожидаемый результат | Полученный результат |
|---|---|---|
| Человеку надо мало:<br>после грома — тишину.<br>Голубой клочок тумана.<br>Жизнь — одну. И смерть — одну.<br>10 | Человеку надо<br>мало: после<br>Грома — тишину.<br>Голубой клочок<br>тумана. Жизнь<br>— одну. И смерть<br>— одну. | Человеку надо<br>мало: после<br>Грома — тишину.<br>Голубой клочок<br>тумана. Жизнь<br>— одну. И смерть<br>— одну. |
| gggggg<br>10 | gggggg | gggggg |
| f d f g d s d f s d f g s a<br>8 | f d f g d<br>s d f s d<br>f g s a | f d f g d<br>s d f s d<br>f g s a |

## Контрольные вопросы.

1) Что такое «конвенции о связи»? Перечислите конвенции, которые вы знаете, и уточните их содержание.

Конвенция о связи определяет порядок передачи параметров в подпрограмму. Конвенция **паскаль** предполагает, что параметры помещаются в стек в прямом порядке, то есть в том же порядке, как они записаны в списке параметров. Конвенции **си, стандартная и защищённая** предполагают обратный порядок записи параметров в стек. **Регистровая** конвенция позволяет передавать до трёх параметров через регистры.

2) Каково внутреннее представление данных в программе на Паскале? На C++?

В программах на C++ и Паскаль данные во внутреннем представлении выглядят как последовательность байт. В зависимости от объявленного типа данных элемент в памяти может занимать разное количество байт. Например, элементы типа real в Паскаль занимаю 6 байт, а int – 4 байта, в C++ для данных типа int и float отводится по 4 байта в памяти.

3) Какие особенности компоновки необходимо учитывать при написании модуля на ассемблере?

При написании модуля на ассемблере нужно учитывать:

- особенности формирования внутренних имен;
- соответствия типов данных;
- что модуль на ассемблере должен содержать пролог и эпилог;
- что регистры EBX, EBP, ESI, EDI нужно сохранять.

4) Какие конвенции вы использовали при создании своей программы?

В своей работе я использовала конвенцию **си**. Запись параметров в стек при этом обратная, а очистку стека осуществляет вызывающая подпрограмма.

5) Как связана структура данных стека в момент передачи управления и текст программы и подпрограмм?

В момент передачи управления в стеке находятся передаваемые параметры и адрес возврата, а во время работы подпрограммы в стек добавляется старое значение регистра EBP и область локальных переменных.

6) С какой целью применяют разноязыковые модули в одном проекте?

Разноязыковые модули в одном проекте создаются обычно для экономии памяти и времени. Ассемблер позволяется напрямую работать с памятью и процессором напрямую, поэтому почти не тратится зря процессорное время.

**Вывод.**

Я узнала, что такое конвенции о связях и какие они бывают, чем отличаются и какие имею особенности. Изучила особенности написания

модулей на ассемблере, которые нужно учитывать при работе с разноязыковыми модулями. В ходе выполнения домашней работы я разработала схемы алгоритмов каждого модуля, создала консольное приложение с разноязыковыми модулями в Visual Studio и скомпоновала части программы в единый проект средствами VS. Основной модуль написала на C++, он вызывает подпрограмму на ассемблере, которая разбивает текст на строки в соответствии в заданной длиной, а затем вызывает подпрограмму на C++, которая выравнивает правую границу строк.