ГУАП
КАФЕДРА №14

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

		Коренева Е.А
Должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ

по курсу: ПРОГРАММИРОВАНИЕ НА ЯЗЫКАХ АССЕМБЛЕРА

РАБОТУ ВЫПОЛНИЛ
СТУДЕНТ ГР. 1441

подпись, дата	инициалы, фамилия

Санкт-Петербург
2016

Оглавление

1. Постановка задачи

Реализовать игру “виселица”, используя для строковых операций ассемблер x86

2. Формализация задачи

Программа должна принимать ввод игрока, выводить превдографическое изображение виселицы, отображающее текущее состояние игры и строку, состоящую из пробелов и угаданных игроком букв.

3. Алгоритм [2]

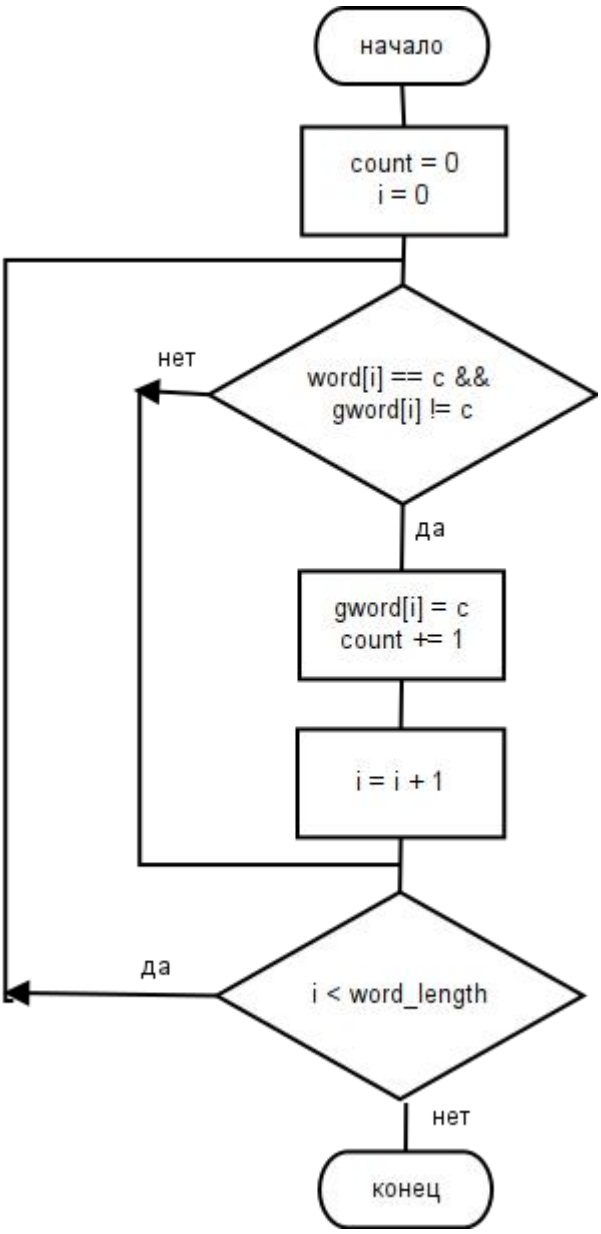

Рисунок 1. Схема алгоритма

4. Листинг [1]

```
#include <stdio.h>
```

```c
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include "text.h"

#define HISTORY_LEN 16

typedef enum { GAME, WON, FAIL } state_t;

void draw(const char* str) {
    printf("%s\n", str);
}

char ask_user(char* history) {
    static int history_index = 0;

    char c;
    printf("Enter letter: ");
    do {
        scanf("%c", &c);
    } while(!isalpha(c));
    printf("\n");


    history[history_index++ % (HISTORY_LEN-1)] = c;

    return c;
}

int update_word(const char* word, char* game_word, char c) {
    int count = 0;

    asm __volatile__ (
    "    mov eax, 0;"           // обнуляем счетчик букв
    "    mov rbx, %1;"          // копируем адрес word
    "    mov rcx, %2;"          // копируем адрес game_word
    "loop1:"                    // метка цикла
    "        cmp [rbx], %3;"    // сравниваем букву по указателю word с введенной
    "        jne next;    "     // если не совпадает, переходим к следующей итерации цикла
    "        cmp [rcx], %3;"    // сравниваем букву по указателб на game_word с введенной
    "       je  next;    "    // если совпадает (буква уже была введена), переходим к следующей итерации
    "        inc eax;"          // увеличиваем смктчик угаданных букв
    "        movb [rcx], %3;"   // копируем букву в game_word по текущему указателю
    "    next:"                 // метка перехода к следующей итерации
    "        inc rbx;"          // перемещаемся к следующей букве в word
    "        inc rcx;"          // перемещаемся к следующей букве в game_word
    "        cmpb [rbx], 0;"    // сравниваем текущую букву с символом конца строки
    "        jne loop1;"        // если не конец строки, переходим  начало цикла
    "    mov %0, eax;"          // копируем счетчик букв в переменную
    : "=r" (count)
    : "r" (word), "r" (game_word), "r" (c)
    : "rax", "rbx", "rcx"
    );

    return count;
}

int main(int argc, char *argv[]) {
    const char* word = "assembler";
```

```
    char* game_word  = (char*) malloc(word);
    char history[HISTORY_LEN];
    int level;


    // Заполняем историю и слово пробелами
    memset(game_word, '_', strlen(word) * sizeof(char));
    memset(history, '_', (HISTORY_LEN - 1) * sizeof(char));


    state_t state = GAME;
    char c;
    int changed_letters;

    while(state == GAME) {
        system("clear");
        fflush(stdin);
        fflush(stdout);
        draw(history);
        draw(levels[level]);
        draw(game_word);
        c = ask_user(history);
        changed_letters = update_word(word, game_word, c);
        if(changed_letters == 0) {
            level++;
        }
        if(strcmp(word, game_word) == 0) {
            state = WON;
        }
        if(level >= MAX_LEVEL) {
            state = FAIL;
        }
    }

    if(state == WON) {
        draw(you_won);
    } else {
        draw(game_over);
    }

    return 0;
}
```

5. Источники

1. x86 Assembly: https://en.wikibooks.org/wiki/X86_Assembly
2. ГОСТ 19.701-90