Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №12 «Компьютерные системы и технологии»

**ОТЧЕТ**

**О выполнении лабораторной работы №6**

**«Работа со структурами данных на основе списков»**

**Cтудент**: Гатченко А.С.

**Группа**: Б22-525

**Преподаватель**: Половнева Ю. А.

**1. Формулировка индивидуального задания**

К каждому слову строки, начинающемуся на согласную букву, добавить некоторый константный префикс.

**2. Описание использованных типов данных**

При выполнении данной лабораторной работы использовались встроенные типы данных int и char, предназначенные для работы с целыми числами, символами и строками, а также структуры данных.

**3. Описание использованного алгоритма**

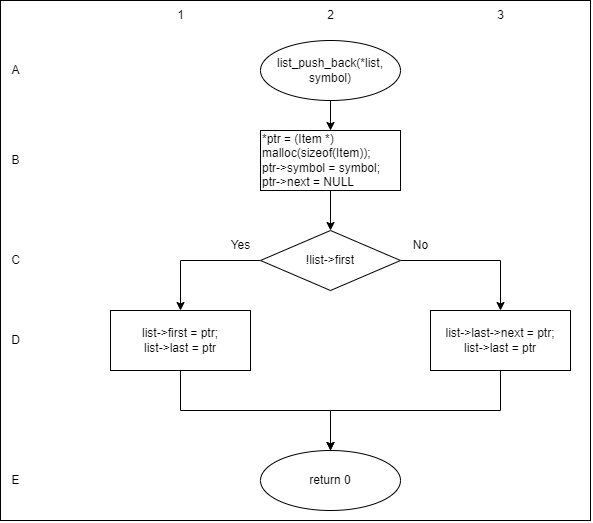
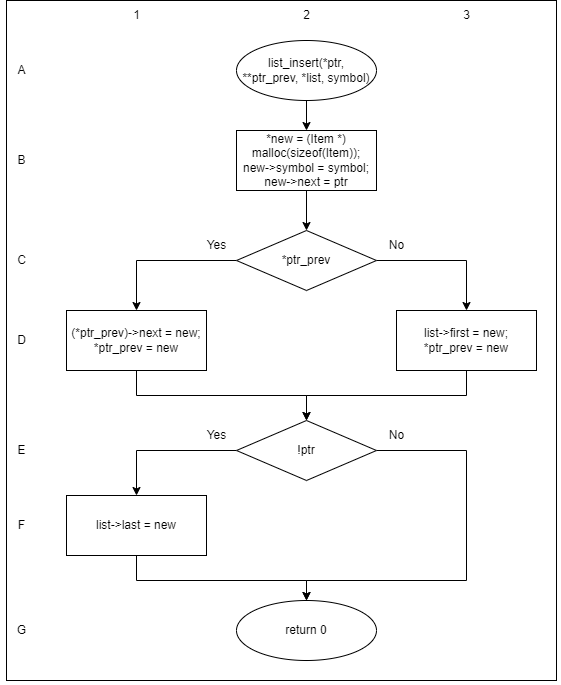


Рис. 1: Блок­схема алгоритма работы функции list\_push\_back()



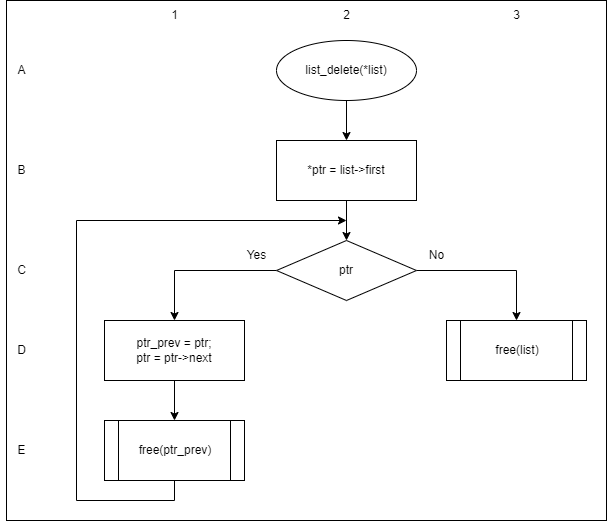


Рис. 2: Блок-схема алгоритма работы функции list\_insert()

Рис. 3: Блок-схема алгоритма работы функции list\_delete()

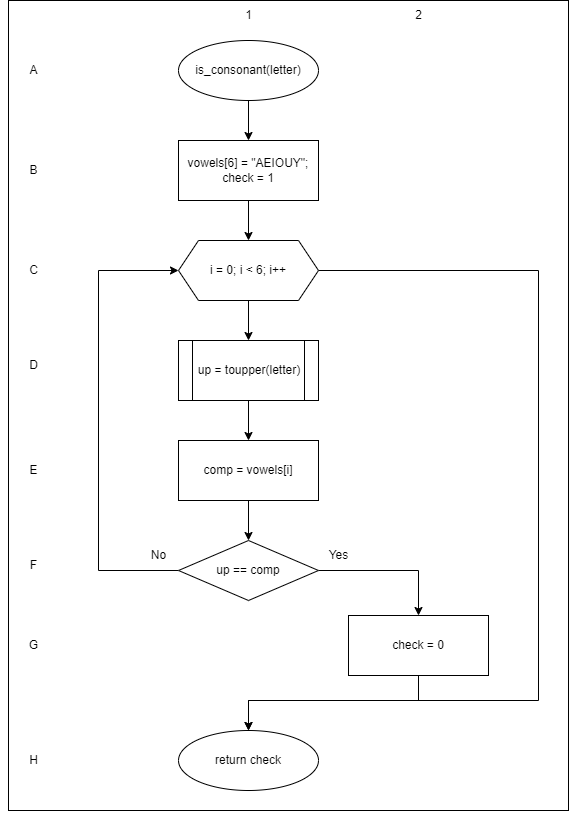
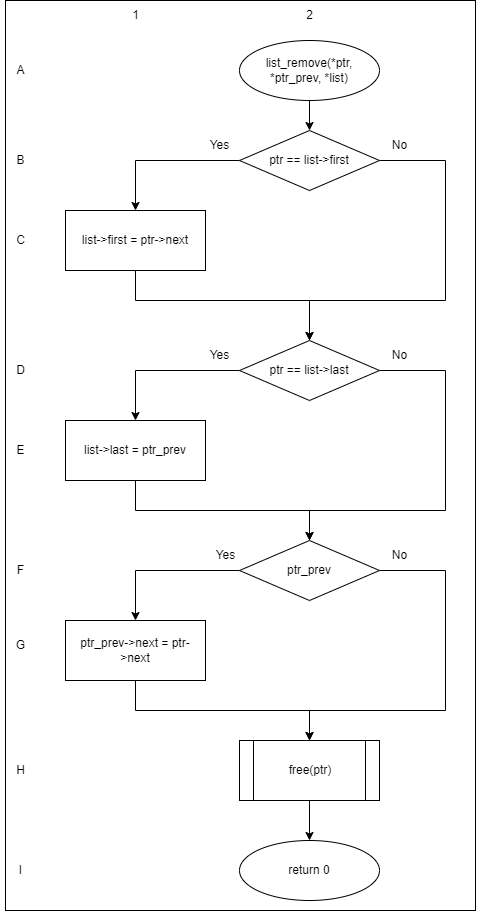


Рис. 4: Блок-схема алгоритма работы функции is\_consonant()

Рис. 5: Блок-схема алгоритма работы функции list\_remove()

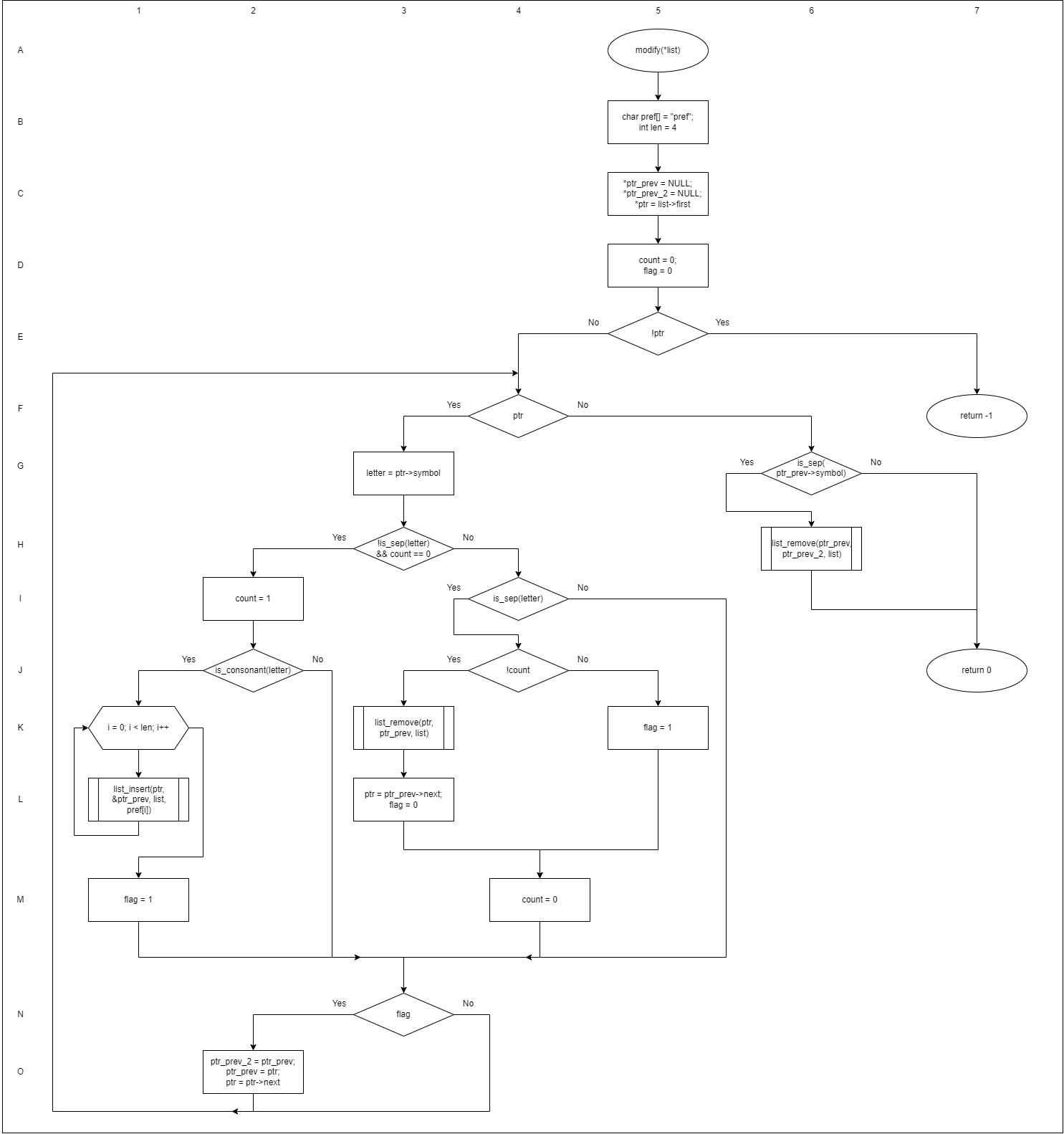


Рис. 6: Блок-схема алгоритма работы функции modify()

**4. Исходные коды разработанных программ**

Листинг 1: Исходные коды программы lab6

1) Файл: prog.c

#include <stdio.h>  
#include "list.h"  
  
int input(List \*list){  
 printf("Input some words:\n");  
 int letter = getchar();  
  
 while (letter == ' ' || letter == '\t'){  
 letter = getchar();  
 }  
  
 while (letter != '\n' && letter != EOF){  
 list\_push\_back(list, letter);  
 letter = getchar();  
 }  
 if (letter == EOF){  
 return -1;  
 }  
 return 0;  
}  
  
int main(){  
 List \*list = list\_new();  
 int check = input(list);  
  
 while (check != -1){  
 printf("Initialized list:\n");  
 list\_print(list);  
  
 printf("New list:\n");  
 modify(list);  
 list\_print(list);  
  
 list\_delete(list);  
 list = list\_new();  
 check = input(list);  
 }  
  
 list\_delete(list);  
 return 0;  
}

2) Файл: list.c

#include <stdlib.h>  
#include <stdio.h>  
#include <ctype.h>  
#include "list.h"  
  
  
List \*list\_new(){  
 return (List \*) calloc(1, sizeof(List));  
}  
  
void list\_print(const List \*list){  
 Item \*ptr = list->first;  
 printf("'");  
 while (ptr){  
 printf("%c", ptr->symbol);  
 ptr = ptr->next;  
 }  
 printf("'");  
 printf("\n");  
}  
  
int list\_push\_back(List \*list, char symbol){  
 Item \*ptr = (Item \*) malloc(sizeof(Item));  
 ptr->symbol = symbol;  
 ptr->next = NULL;  
  
 if (!list->first){  
 list->first = ptr;  
 list->last = ptr;  
 }  
 else {  
 list->last->next = ptr;  
 list->last = ptr;  
 }  
 return 0;  
}  
  
int list\_insert(Item \*ptr, Item \*\*ptr\_prev, List \*list, const char symbol){  
 Item \*new = (Item \*) malloc(sizeof(Item));  
 new->symbol = symbol;  
 new->next = ptr;  
  
 if (\*ptr\_prev){  
 (\*ptr\_prev)->next = new;  
 \*ptr\_prev = new;  
 }  
 else {  
 list->first = new;  
 \*ptr\_prev = new;  
 }  
 if (!ptr){  
 list->last = new;  
 }  
 return 0;  
}  
  
int list\_insert\_chunk(Item \*ptr, Item \*\*ptr\_prev, List \*list, char \*text, int len){  
 List \*new\_list = list\_new();  
  
 for (int i = 0; i < len; i++){  
 list\_push\_back(new\_list, text[i]);  
 }  
  
 if (!(\*ptr\_prev)){  
 \*ptr\_prev = new\_list->first;  
 list->first = \*ptr\_prev;  
 }  
 else {  
 (\*ptr\_prev)->next = new\_list->first;  
 }  
  
 new\_list->last->next = ptr;  
 \*ptr\_prev = new\_list->last;  
 return 0;  
}  
  
int list\_remove(Item \*ptr, Item \*ptr\_prev, List \*list){  
 if (ptr == list->first){  
 list->first = ptr->next;  
 }  
 if (ptr == list->last){  
 list->last = ptr\_prev;  
 }  
 if (ptr\_prev){  
 ptr\_prev->next = ptr->next;  
 }  
 free(ptr);  
 return 0;  
}  
  
int is\_sep(const char letter){  
 return (letter == ' ' || letter == '\t');  
}  
  
int is\_consonant(const char letter){  
 char vowels[6] = "AEIOUY";  
 int check = 1;  
 for (int i = 0; i < 6; i++){  
 char up = toupper(letter);  
 char comp = vowels[i];  
  
 if (up == comp){  
 check = 0;  
 break;  
 }  
 }  
 return check;  
}  
  
int modify(List \*list){  
 char pref[] = "pref";  
 int len = 4;  
 Item \*ptr\_prev = NULL;  
 Item \*ptr\_prev\_2 = NULL;  
 Item \*ptr = list->first;  
 int count = 0;  
 int flag = 0;  
 char letter;  
 if (!ptr){  
 return -1;  
 }  
 while (ptr){  
 letter = ptr->symbol;  
 if (!is\_sep(letter) && count == 0){  
 count = 1;  
 if (is\_consonant(letter)){  
 #ifdef chunk  
 list\_insert\_chunk(ptr, &ptr\_prev, list, pref, len);  
 #else  
 for (int i = 0; i < len; i++){  
 list\_insert(ptr, &ptr\_prev, list, pref[i]);  
 }  
 #endif  
 }  
 flag = 1;  
 }  
 else if (is\_sep(letter)){  
 if (!count){  
 list\_remove(ptr, ptr\_prev, list);  
 ptr = ptr\_prev->next;  
 flag = 0;  
 }  
 else {  
 flag = 1;  
 }  
 count = 0;  
 }  
 if (flag){  
 ptr\_prev\_2 = ptr\_prev;  
 ptr\_prev = ptr;  
 ptr = ptr->next;  
 }  
 }  
 if (is\_sep(ptr\_prev->symbol)){  
 list\_remove(ptr\_prev, ptr\_prev\_2, list);  
 }  
 return 0;  
}  
  
void list\_delete(List \*list){  
 Item \*ptr = list->first, \*ptr\_prev;  
 while (ptr){  
 ptr\_prev = ptr;  
 ptr = ptr->next;  
 free(ptr\_prev);  
 }  
 free(list);  
}

3) Файл: list.h

#ifndef LIST\_H  
#define LIST\_H  
  
typedef struct Item {  
 char symbol;  
 struct Item \*next;  
} Item;  
  
typedef struct List {  
 Item \*first;  
 Item \*last;  
} List;  
  
List \*list\_new();  
void list\_print(const List \*list);  
int modify(List \*list);  
int list\_push\_back(List \*list, char symbol);  
void list\_delete(List \*list);  
  
#endif

**5. Описание тестовых примеров**

Таблица 1: Тестовые примеры lab6

|  |  |  |
| --- | --- | --- |
| Введенная строка | Ожидаемый результат после обработки | Полученный результат после обработки |
| ' wejkfhwuhwijfkj ik a ldwak ng kesjkfn kj n g efsef ht' | 'prefwejkfhwuhwijfkj ik a prefldwak prefng prefkesjkfn prefkj prefn prefg efsef prefht' | 'prefwejkfhwuhwijfkj ik a prefldwak prefng prefkesjkfn prefkj prefn prefg efsef prefht' |
| 'aaaaaa iiiiii k k klklakd w wa wadfr d ' | 'aaaaaa iiiiii prefk prefk prefklklakd prefw prefwa prefwadfr prefd' | 'aaaaaa iiiiii prefk prefk prefklklakd prefw prefwa prefwadfr prefd' |
| 'jjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj' | 'prefjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj' | 'prefjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjj' |
| 'uooooooouooooooooooooooooooooooooooooooou' | 'uooooooouooooooooooooooooooooooooooooooou' | 'uooooooouooooooooooooooooooooooooooooooou' |

**6. Скриншоты**

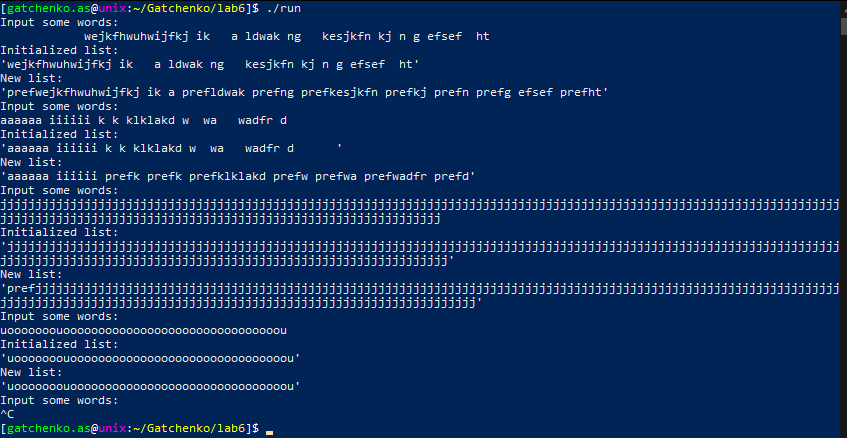


Рис. 7: Работа программы lab6

**7. Выводы**

В ходе выполнения данной работы на примере программы, выполняющей обработку строк на основе списков структур данных, были рассмотрены базовые принципы работы построения программ на языке C, обработки строк и использования структур:

1. Организация ввода/вывода, а также проверка корректности ввода.
2. Разработка функций.
3. Объявление и использование переменных.
4. Выполнение простейших арифметических операций над целочисленными и дробными операндами.
5. Использование циклов и условий.
6. Использование структур данных.
7. Обработка строк.
8. Использование указателей (параметров) на структуры данных.
9. Разбиение программы на несколько файлов.
10. Работа с памятью.