## 0.1 Структуры

**Задача SM.1.** Реализовать структуру «Дробь» - целая часть,числитель, знаменатель. Реализовать функции работы с дробями (дроби-параметры, возвращаемое значение дробь): сложение, вычитание, умножение, деление, печать дроби в виде <целая часть>(числитель/знаменатель) и в виде десятичной дроби с указаниеми периода (Например $\frac{5}{7} = 0,(714285)$)

## 0.2 Массивы символов. Строки

В языке C операции работы со строкой как с единым целым не предусмотрены. Строка — это массив симоволов, заканчивающийся символом “/0”. Определить строки можно различными способами. Определяется массив символов, заведомо известного размера, таким же способом как любой другой массив. Затем, по мере необходимости происходит инициализация элементов этого массива и обращение к ним.

```
 char chararray[10];
chararray[0]='s';
...
```

Можно определить строку, сразу проинициализировав ее, поместив в нее какое– либо значение.

```
 char mystring[]="string"
 \\во внутреннем представлении "stroka\0"
```

В этом случае выделяется память, необходимая для того, чтобы поместить в нее всю последовательность символов и “ 0”. Размер массива определяется количеством введенных символов. mystring указывает на начало массива — выделенной области памяти. В этом случае любой элемент массива может быть изменен. Строку можно определить через указатель.

```
char *pstring="string";
// во внутреннем представлении "строка\0"
printf("%c\n",ms[3]);
```

В этом случае pstring определяется только как указатель на символ. Как каждому указателю ему можно присвоить другое значение, и он будет указывать на что–то другое. Пока же он указывает на строковую константу, которой он инициализирован. К символам этой строки также

можно обращаться как к элементам При попытке изменить константу, результат будет неопределен.

Для работы со строками используются библиотечные функции, описанные в библиотеке string.h или stdlib.h

В процессе работы программы не всегда получается последовательность символов с таким окончанием. Например, при вводе строки введенная строка может оканчиваться символом $\backslash n$ или заполненение элементов массива не обязано предусматривать введение \0. В этом случае, чтобы организовать правильную работу со строками, необходимо обеспечить чтобы последовательность символов оканчивалась \0.

scanf, как правило, всегда преобразует прочитанную строку в "правильную", но чтение в одну переменную происходит только до разделителя: пробела, переноса строки, табулции и т.д.

Если же необходимо прочитать весь текст, скажем, до переноса строки, для этого подходит функция gets().

Рассмотрим программу, вставляющую подстроку в уже имеющуюся строку.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 20
/* Функция ins_str вставляет в строку s подстроку subs перед
элементом с номером before. При этом проверяется, что размер
полученной строки не должен превышать размер массива, объявленного
в вызывающей функции
*/

int ins_str(char *s, int max, int before, char *subs){
/* Определение буфера для хранения строки, равного
   максимально возможному размеру.
   Так как локальные переменные, в том числе и массив
   в функции создается в момент ее вызова, размер массива
   может быть передан в функцию в качестве параметра.
*/

char buf[max];
 /* Пустая строка. Первый элемент равен "\0".*/

 *buf = '\0';
```

```
/* p указывает на элемент, перед которым вставляется подстрока.*/

 char *p = s + before;

/* Обязательная проверка на возможность выхода за границы
имеющегося массива. В случае выхода за границы
результат будет непредсказуемым.
*/
    if (strlen(s) + strlen(subs) <= max){

/* strcpy() копирует в buf часть строки от элемента
     с номером before и до конца строки
*/
     strcpy(buf,p);

/* Строка s "обрывается" на элементе before присваиванием
   этому элементу значения "/\0". *

      *p = '\0';

/* strcat() присоединяет к строке s подстроку sub */

      strcat(s,subs);

/* strcat() присоединяет к строке s подстроку buf */

      strcat(s,buf);

/* При успешной вставке влзвращаем 1 */

     return 1;

   }else{

   /* При неудаче возвращаем 0. */
  return 0;
  }
};

int main() {
/*
```

```
  При определении массива его размер должен быть
  известен. В основной программе или при описании
  глобальных переменных размер статического массива
  не может быть параметром.
  MAX - синоним числа 20
*/
 char z[MAX] = "ttatttttt";
  char *zs = "uuu";
/* Вставляем в строку z подстроку zs перед элементом
   с номером 2.
   Проверяется возможность "переполнения" при вставке.
*/

  if ( ! ins_str(z, MAX, 2, zs)) printf("Can't␣insert\n");
  else printf("%s\n",z);

return 0;
}
```

### 0.2.1 Задачи

**Задача SM.2.** Дана строка символов. Слово - непрерывная последовательность букв латинского алфавита. Слова могут быть разделены пробелами (один или больше), запятыми и точками.

Пример строки: ' The cat is animal ., It is,the.dog'

Написать программу, которая подсчитывает количество слов (strstr).

**Задача SM.3.** Дана строка символов. Слово - непрерывная последовательность букв латинского алфавита. Слова могут быть разделены пробелами (один или больше).

Написать программу, которая подсчитывает частоту встречающихся слов (структуры, strstr)

**Задача SM.4.**

**Дано:** Цепочка из бусинок может считаться бусами, только если:

- цепочка состоит из двух симметричных половинок (например: ABCBA, ABBA)
- цепочка состоит повторяющихся периодичных последовательностей бусинок

**Написать программу,** , которая определяет, является ли данная цепочка бусинок бусами.

**Задача SM.5.** В строке A(не более 255 символов) может встретиться подстрока B (размер B не более размера A). Заменить все вхождения B на строку C (размер не более 255 символов). Замена начинается слева. Например, строка A: FFWWWFFSSSSF, строка B: SS, строка C: ZZZ Результирующая строка:FFWWWFFZZZZZZF (strstr, strcat,strcpy);

**Задача SM.6.** В строку записано логическое выражение со скобками (! - отрицанеие, and - и, or - или). В выражении используются две переменные A и B. Выражение и логические значения переменных вводятся.

Определить лигическое значение всего выражения (рекурсия, стек)