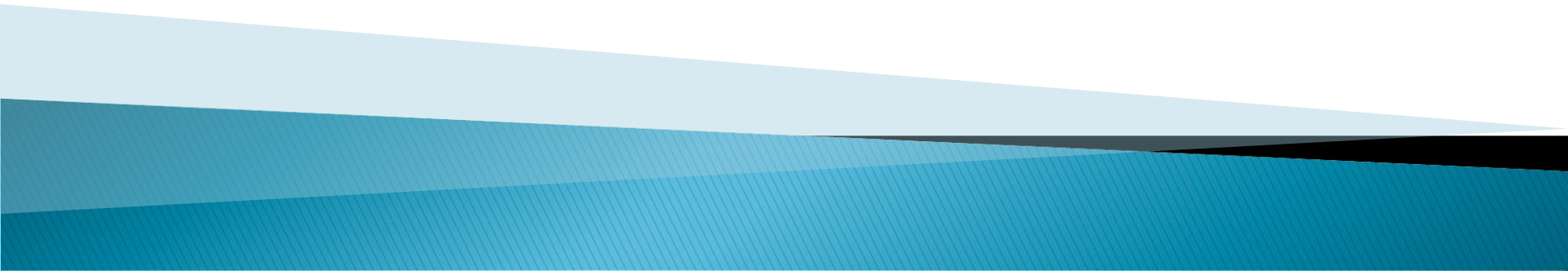


Саркисян Г.Ф.

ПРОГРАММИРОВАНИЕ

Часть I: Структурное программирование

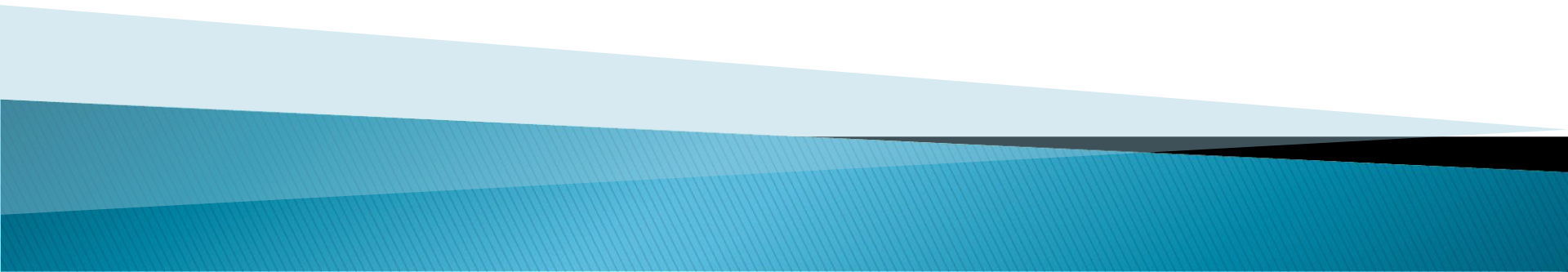


СИМВОЛЬНЫЕ СТРОКИ

Вопросы:

- 1. Определение и инициализация символьной строки*
- 2. Ввод и вывод символьных строк*
- 3. Передача строки в функцию*
- 4. Библиотечные функции работы со строками*
- 5. Массив строк*

ОПРЕДЕЛЕНИЕ и ИНИЦИАЛИЗАЦИЯ СИМВОЛЬНЫХ СТРОК



ОПРЕДЕЛЕНИЕ и ИНИЦИАЛИЗАЦИЯ СИМВОЛЬНЫХ СТРОК

Символьная строка – это массив символов, заканчивающийся нулевым символом.

Нулевой символ – это символ с кодом 0, что записывается следующим образом ‘\0’.

Положение нулевого символа определяет фактическую длину строки. Количество элементов в символьной строке на 1 больше, чем изображение строки.

Спецификация для символьных строк – %s

ОПРЕДЕЛЕНИЕ и ИНИЦИАЛИЗАЦИЯ СИМВОЛЬНЫХ СТРОК

Определение символьной строки:

```
char StringID [SIZE] ;
```

StringID – это указатель на первый символ в строке, т.е. адрес первого символа.

SIZE –1 – это максимальное количество символов, которые могут быть записаны в строке.

Если задан размер массива, а строка короче, то лишние элементы массива содержат так называемый “мусор”.

ОПРЕДЕЛЕНИЕ и ИНИЦИАЛИЗАЦИЯ СИМВОЛЬНЫХ СТРОК

Необходимо запомнить, что символьная константа 'a' и строковая константа "a", содержащая один символ *не одно и тоже!!!*

т. е. 'a' != "a", т.к. "a" который кроме символа 'a' содержит еще и символ '\0'.

Т.к. `sizeof('a') == 1`

`sizeof("a") == 2`

ОПРЕДЕЛЕНИЕ и ИНИЦИАЛИЗАЦИЯ СИМВОЛЬНЫХ СТРОК

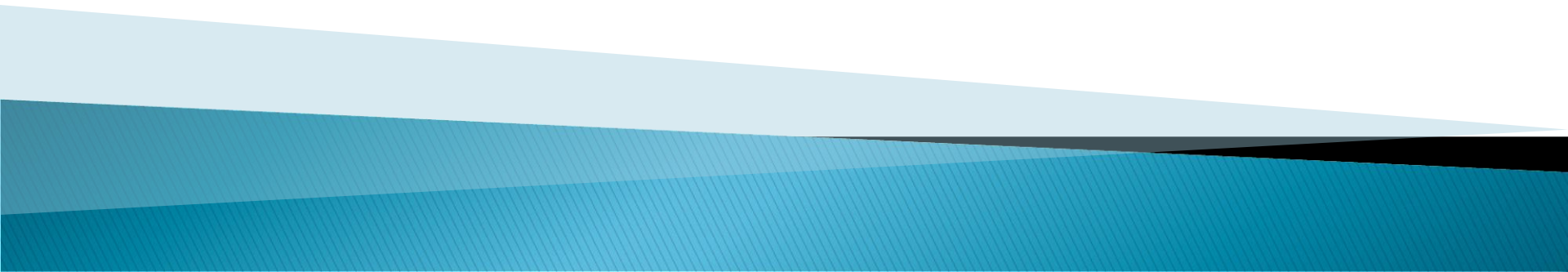
СТРОКИ МОГУТ БЫТЬ ПРОИНИЦИАЛИЗИРОВАНЫ следующими способами:

1. `char str[] = "black";`

2. `char str[] = " 'b' 'l' 'a' 'c' 'k' '\0' ";`

3. `char *str = "black";`

ВВОД и ВЫВОД СИМВОЛЬНЫХ СТРОК



ВВОД и ВЫВОД СИМВОЛЬНЫХ СТРОК

1. Функция `scanf ("%s", StringID) ;`

Единственный случай, когда в функции `scanf` не записывается адрес перед вводимой переменной – это при вводе символьной строки, т.к. имя строки – это и есть адрес.

Для функции `scanf` ввод завершается либо символом пробел, либо символом `'\n'`, причем сами эти символы в строку не заносятся, а заменяются на символ конец строки.

ВВОД и ВЫВОД СИМВОЛЬНЫХ СТРОК

Следовательно, с помощью функции `scanf` МОЖНО ВВОДИТЬ ТОЛЬКО СЛОВА.

```
char str[80];
```

```
scanf("%s", str);
```

```
printf("%s\n", str);
```

ВВОД и ВЫВОД СИМВОЛЬНЫХ СТРОК

2. `char* gets(char*s)` – считывает только строку `s` из стандартного потока.

Для функции `gets` ввод завершается символом `'\n'`, сам символ `'\n'` в строку не заносится, а заменяется на символ конец строки.

Следовательно, с помощью функции `gets` можно вводить предложения

```
char str[80];  
    gets(str);  
    printf("%s\n",str);
```

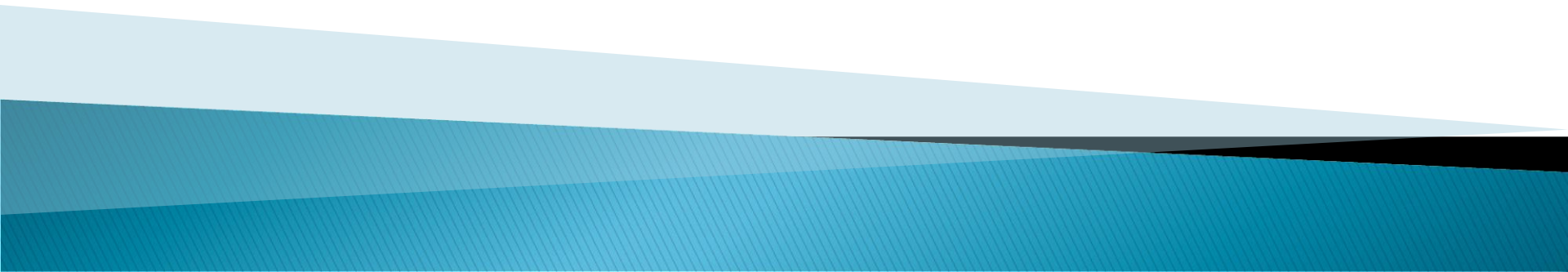
ВВОД и ВЫВОД СИМВОЛЬНЫХ СТРОК

3. `int puts(const char* s)` – записывает строку в стандартный поток, добавляя в конец строки символ `'\n'`.

В случае удачного завершения возвращает значение больше или равное 0 и отрицательное значение (`EOF=-1`) в случае ошибки.

```
char str[80];  
    gets(str);  
    puts(str);  
    puts("Hello!!!!");
```

ПЕРЕДАЧА СТРОКИ В ФУНКЦИЮ



ПЕРЕДАЧА СТРОКИ В ФУНКЦИЮ

Строки при передаче в функции могут передаваться как одномерные массивы типа **char** или как указатели типа **char***.

1. `ReturnType FunctionID(char []);`
2. `ReturnType FunctionID (char *);`

ПЕРЕДАЧА СТРОКИ В ФУНКЦИЮ

```
#include <stdio.h>

int find(char*, char);

void main(void) {
    char gl[7] = "aouiey";
    char str[80];    int i, k;
    gets(str);

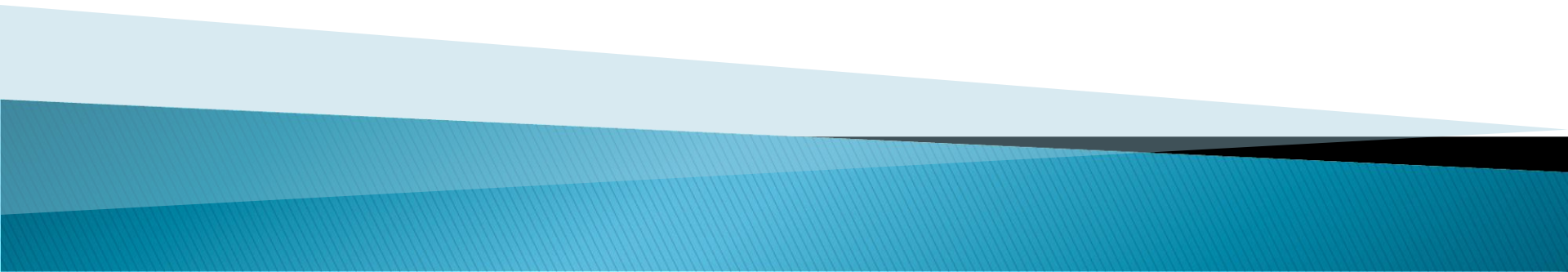
    for(i = 0, k = 0; str[i] != '\0' ; ++i)
        if(find(gl, str[i])>=0) ++k;

    printf("\n%d", k);
}
```

ПЕРЕДАЧА СТРОКИ В ФУНКЦИЮ

```
int find(char* s, char c) {  
    for (int i = 0; s[i] ; ++i)  
        if(s[i] == c)  
            return i;  
  
    return -1;  
}
```


БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ



БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

Для работы со строками существуют специальные библиотечные функции, которые содержатся в заголовочном файле **string.h**

1. Функция присваивания:

```
char* strcpy (char*s1, const char*s2);
```

копирует строку *s2* в строку *s1*, причем размер строки *s1* должен быть достаточно большим, чтобы хранить строку *s2* и символ конец строки.

Возвращает значение строки *s1*.

БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

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

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

```
void main(void) {
```

```
    char str1[80]= "bbb ccc ddd", str2[80];
```

```
    strcpy(str2, str1);
```

```
    puts(str2);
```

```
    strcpy(str1, "aaaaa");
```

```
    puts(str1);
```

```
}
```

БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

2. 1. Функция добавления:

```
char* strcat (char* s1, const char* s2) ;
```

добавляет к строке *s1* строку *s2*, причем размер строки *s1* должен быть достаточно большим, чтобы хранить строки *s1* и *s2*, а так же символ конец строки.

Возвращает значение строки *s1*.

БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

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

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

```
void main(void) {
```

```
    char str1[80]= "bbb ccc ddd";
```

```
    char str2[80]="123 456 7";
```

```
    strcat( strcat(str1,str2), "!" );
```

```
    puts(str1);
```

```
}
```

БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

2.2. Функция добавления:

```
char* strncat(char*s1,const char*s2,int n) ;
```

добавляет не более **n** символов из строки **s2** в строку **s1**, причем если **n** больше, чем длина строки **s2**, то переписывает всю строку **s2**.

Возвращает значение строки s1.

БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

3.1. Функция сравнения:

```
int strcmp (const char*s1, const char*s2) ;
```

Сравнивает посимвольно строки *s1* и *s2*, *причем регистр учитывается.*

Возвращает:

- 0 если строки одинаковы;
- 1 если строка *s1*, больше, чем строка *s2*;
- 1 если строка *s1*, меньше, чем строка *s2*.

БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

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

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

```
void main(void) {
```

```
    char str1[100]="Hello", str2[80]="hello";
```

```
    int k;
```

```
    k = strcmp(str1, str2);
```

```
    if( !k ) puts("The same.");
```

```
    else printf("k = %d\n", k);
```

```
}
```


БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

3.2. Функция сравнения:

```
int strncmp (const char*s1, const char*s2,  
              int n);
```

Сравнивает не более n символов , *причем регистр учитывается.*

```
char str1[100]="Hello";  
char str2[80]="Hello, Dima";  
int k;  
k = strncmp(str1, str2, 5);  
if( !k ) puts("The same.");  
else printf("k = %d\n",k);
```

БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

3.3. Функция сравнения:

```
int strcmpi (const char*s1, const char*s2) ;
```

В данной функции регистр не учитывается.

```
char str1[100]="Hello", str2[80]="hello";
```

```
int k;
```

```
k = strcmpi(str1, str2) ;
```

```
if( !k ) puts("The same.") ;
```

```
else printf("k = %d\n", k) ;
```

БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

4. Функция определяющая фактический размер строки:

```
int strlen (const char*s) ;
```

Возвращает количество символов, предшествующих символу конец строки.

```
char str[80] ;  
int len ;  
gets (str) ;  
len = strlen (str) ;  
printf ("%s\tdlina=%d\n", str, len) ;
```

БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

5. Функция выполняет поиск первого вхождения подстроки `s2` в строку `s1`.

```
char* strstr(const char*s1, const char*s2) ;
```

Если подстрока `s2` содержится в `s1`, то функция возвращает адрес символа из строки `s1`, с которой начинается подстрока `s2`.

Если же такой подстроки нет, то возвращает `NULL`.

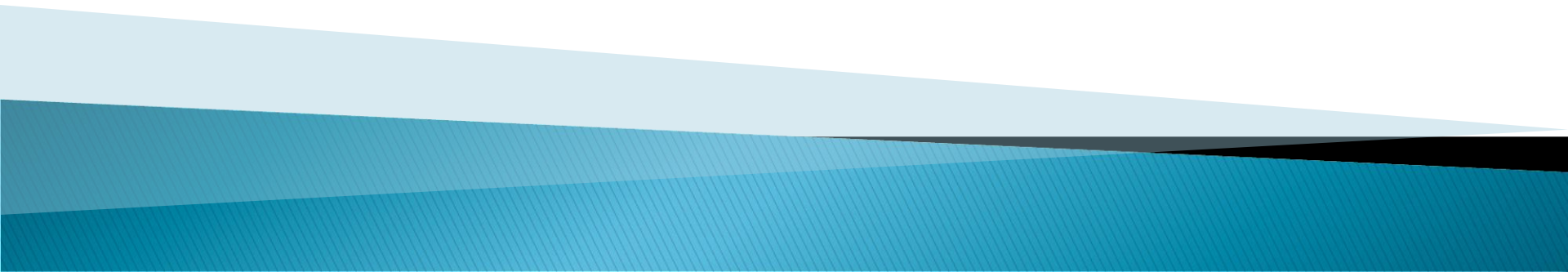
БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

```
char x[100], y[100], *ptr;
int kol = 0;
puts("Введите строку: ");    gets(x);
puts("Введите слова для поиска в строке:");
scanf("%s", y);
ptr = strstr(x, y);
while(ptr) {
    ++kol;
    ptr = strstr(ptr += strlen(y), y);
}
printf("k=%d\n", kol);
```

БИБЛИОТЕЧНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

```
char text[500];
int i, j, arr[255]={0};
printf("\nВведите текст:\n"); gets(text);
for(j = 0;text[j]; ++j){
    i = text[j];
    ++arr[i];
}
for(i = 0;i < 255; ++i)
    if(arr[i])
        printf("\nСимвол '%c' встречается %d раз",
                i,arr[i]);
```

МАССИВ СТРОК



МАССИВ СТРОК

Определение массива строк

```
char StringID [Size1][Size2];
```

Size1– это целая положительная константа, т.е. количество указателей, которые содержат адреса строк.

Size2 – размер каждой строки.

StringID – это адрес адреса, т.е. указатель на указатель.

МАССИВ СТРОК

Ввод и вывод массива строк

```
char str[5][20];
```

```
int i;
```

```
// ввод массива строк
```

```
for(i = 0; i < 5; ++i) {  
    printf("Строка %d\n", i+1);  
    gets(str[i]);  
}
```

```
// вывод массива строк
```

```
for(i = 0; i < 5; ++i)    puts(str[i]);
```

АЛГОРИТМ МЕТОД ПРОСТОГО ВЫБОРА

```
for( i = 0; i < n - 1; ++i) {  
    k = i;  
    for( j = i + 1; j < n; ++j)  
        if (x[k] > x[j])  
            k = j;  
    buf = x[i];  
    x[i] = x[k];  
    x[k] = buf;  
}  
  
char x[5][20];
```

MACCIB CTPOK

```
char x[5][20], buf[20];
int i, j, k;
for( i = 0; i < 4 ; ++i){
    k = i;
    for(j = i+1; j < 5; ++j)
        if(strcmp( x[k], x[j] ) > 0)
            k=j;

    strcpy(buf, x[i]);
    strcpy(x[i], x[k]);
    strcpy(x[k], buf);
}
```

СПАСИБО ЗА ВНИМАНИЕ!

ВОПРОСЫ ?

СИМВОЛЬНЫЕ СТРОКИ

Автор: Саркисян Гаяне Феликсовна