

# Лекция 4

Алгоритмические языки и  
программирование

# Часть 1

# Указатели

- Указатель — это переменная, содержащая адрес ячейки памяти (числовое значение).
- Память типичной машины представляет собой массив последовательно пронумерованных(адресованных) ячеек, с которыми можно работать по отдельности или в виде массива.
- Синтаксис объявления указателей:
- `<тип> *<имя>;`
- Например:  
`float *a;`  
`long int *b;`
- Два основных оператора для работы с указателями – это оператор `&` взятия адреса, и оператор `*` разыменования.

# Указатели и адреса

- Оператор **&**(взятие адреса) применяется только к “объектам”, расположенным в памяти: к переменным и элементам массивов.
- Унарный оператор **\***(разыменование) есть оператор косвенного доступа. Примененный к указателю, он выдает “объект”, на который данный указатель указывает.

# Пример

```
#include <conio.h>
```

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

```
void main() {
```

```
    int A = 100;
```

```
    int *p;
```

```
    p = &A; //Получаем адрес переменной A
```

```
    printf("%p\n", p); //Выводим адрес переменной A
```

```
    printf("%d\n", *p); //Выводим содержимое переменной A
```

```
    *p = 200; //Меняем содержимое переменной A
```

```
    printf("%d\n", A);
```

```
    printf("%d", *p);
```

```
}
```

# Перестановка двух переменных

```
void swap(int x, int y) /* НЕВЕРНО
*/
{
    int temp;
    temp = x;
    x = y;
    y = temp;
}
```

# Перестановка двух переменных

- Чтобы получить желаемый эффект, вызывающей программе надо передать указатели на те значения, которые должны быть изменены:

```
swap(&x, &y);
```

# Перестановка двух

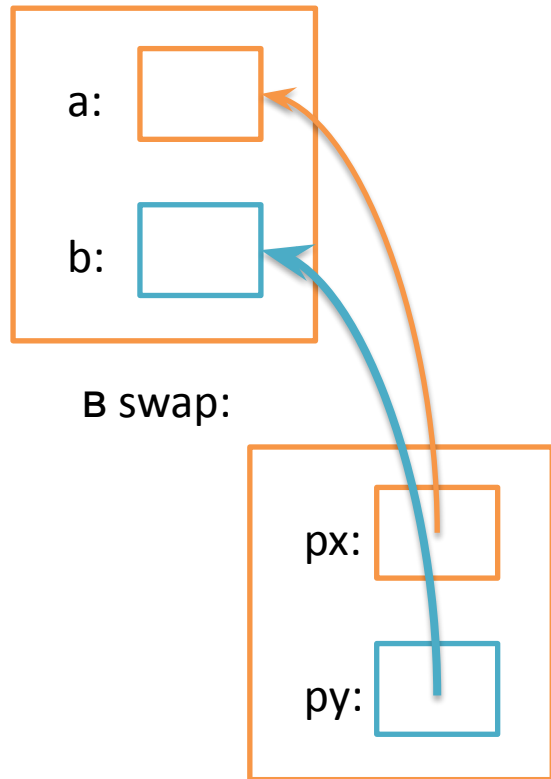
переменных

```
void swap(int *px, int *py)
{
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}
```



# Перестановка двух переменных

- Аргументы-указатели позволяют функции осуществлять доступ к объектам вызвавшей ее программы и дают возможность изменить эти объекты.

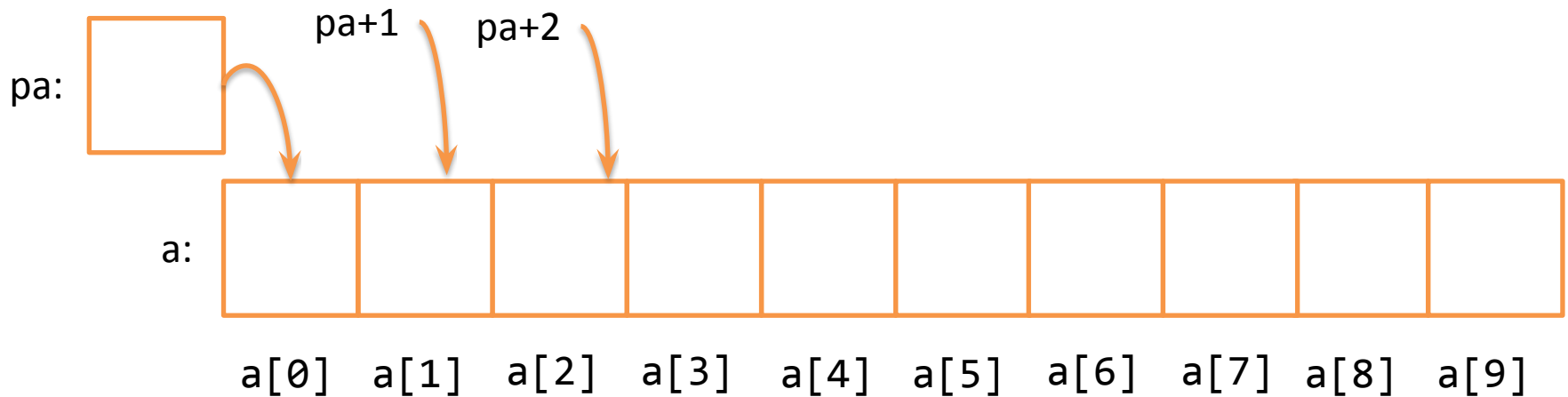


# Часть 2

# Адресная арифметика

- Указатели и целочисленные переменные можно складывать и вычитать. Конструкция  $p + n$  означает адрес объекта, занимающего  $n$ -е место после объекта, на который указывает  $p$ . Это справедливо безотносительно к типу объекта (исключение `void`), на который указывает  $p$ ;  $n$  автоматически домножается на коэффициент, соответствующий размеру объекта. Информация о размере неявно присутствует в объявлении  $p$ . Если, к примеру, `int` занимает четыре байта, то коэффициент умножения будет равен четырем.

# Указатели и массивы



```
int a[10];
```

```
int *pa = NULL;
```

```
pa = &a[0]; /* будет указывать на нулевой элемент a,  
иначе говоря, pa будет содержать адрес элемента a[0].*/
```

```
x = *pa; // копирует содержимое a[0] в x.
```

```
*(pa+1); // возвращает первый элемент массива
```

# Нулевой элемент массива и адрес

```
/* ра и а имеют одно и то же значение.  
*/
```

```
ра = &a[0];
```

```
/* Поскольку имя массива является  
синонимом расположения его  
начального элемента, присваивание ра=&a  
[0] можно также записать в следующем  
виде: */
```

```
ра = а;
```

```
// а[i] можно записать как *(а+i)
```

# Адресная арифметика

Важно помнить что следующие операции опасны:

- Использовать арифметические операции с указателями ссылающимися не на массив.
- Арифметические операции между указателями на разные массивы.
- Выход за пределы массива(начало и конец) используя адресную арифметику.

# Длина строки

```
/* strlen: возвращает длину строки */  
int strlen(char *s)  
{  
    int n;  
    // увеличение на 1 некоторой копия  
указателя, находящегося в личном  
пользовании функции strlen.  
    for (n = 0; *s != '\0' ; s++)  
        n++;  
    return n;  
}
```

# Длина строки

```
/* все вызовы правомерны */  
strlen("Здравствуй, мир"); /* строковая константа */  
char array[100];  
strlen(array); /* char array[100]; */  
char * prt = NULL;  
strlen(ptr); /* char *ptr; */
```



# Длина строки

```
/* strlen: возвращает длину строки s */  
int strlen(char *s)  
{  
    char *p = s;  
    while (*p != '\0' )  
        p++;  
    return p - s;  
}
```

# Символы и строки в С

Функция	Пояснение
<a href="#"><u>strlen</u></a> (имя_строки)	определяет длину указанной строки, без учёта нуль-символа
<b>Копирование строк</b>	
<a href="#"><u>strcpy</u></a> (s1,s2)	выполняет побайтное копирование символов из строки s2 в строку s1
<b>Конкатенация строк</b>	
<a href="#"><u>strcat</u></a> (s1,s2)	объединяет строку s2 со строкой s1. Результат сохраняется в s1
<a href="#"><u>strncat</u></a> (s1,s2,n)	объединяет n символов строки s2 со строкой s1. Результат сохраняется в s1
<b>Сравнение строк</b>	
<a href="#"><u>strcmp</u></a> (s1,s2)	сравнивает строку s1 со строкой s2 и возвращает результат типа int: 0 –если строки эквивалентны, >0 – если s1<s2, <0 — если s1>s2 С учётом регистра

# Символы и строки в С

## Функции поиска

### [strchr\(s,c\)](#)

поиск первого вхождения символа с в строке s. В случае удачного поиска возвращает указатель на место первого вхождения символа с. Если символ не найден, то возвращается ноль.

### [strstr\(s1,s2\)](#)

Возвращает указатель первого вхождения любого символа строки s2 в строке s1, или пустой указатель, если строка s2 не является частью строки s1

## Функции стандартной библиотеки ввода/вывода <stdio>

### [getchar\(c\)](#)

считывает символ с со стандартного потока ввода, возвращает символ в формате int

### [gets\(s\)](#)

считывает поток символов со стандартного устройства ввода в строку s до тех пор, пока не будет нажата клавиша ENTER

Подробнее на сайте:

<http://cppstudio.com/post/437/>

# Лабораторные работы

# Указатели

- Создайте и заполните массив из 10 элементов, числами от 100 до 110. Напишите программу, которая будет выводить адреса элементов массива. Проанализируйте как меняются адреса элементов массива.
  - Примечание:
    1. Использовать циклы;
    2. Использовать указатели;

# Строки

- Напишите программу, вычисляющую количество символов в строке.

Примечание:

1. Использовать указатели
2. Использовать циклы

# Среднее арифметическое последовательности чисел

- Напишите функцию для нахождения среднего арифметического последовательности чисел, если известно, что признак конца списка (цифра '0').

Примечание:

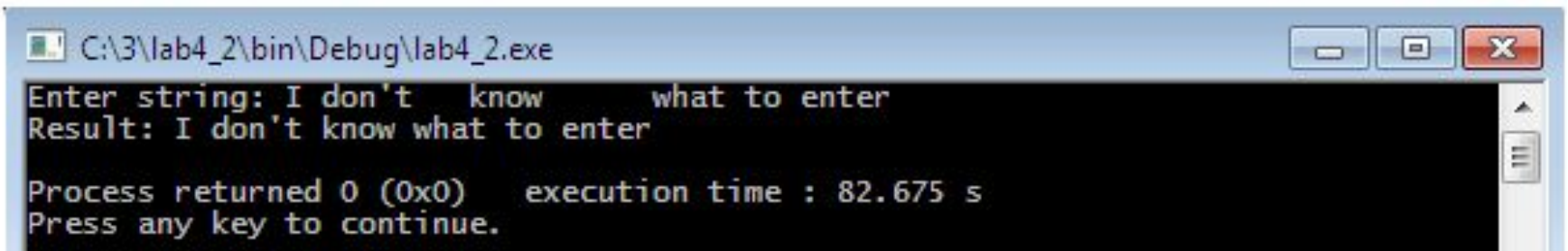
1. Использовать указатели
2. Использовать циклы
3. Использовать функции

# Замена символов

- Дана строка (максимально 100 символов), содержащая слова, разделенные одним или несколькими пробелами, или знаками табуляции. Заменить все знаки табуляции знаком пробела, удалить двойные пробелы из строки. При реализации программы.

Примечание:

1. Использовать функции из библиотеки **string.h**
2. Использовать циклы



```
C:\3\lab4_2\bin\Debug\lab4_2.exe
Enter string: I don't know what to enter
Result: I don't know what to enter

Process returned 0 (0x0)   execution time : 82.675 s
Press any key to continue.
```