

Материалы презентации предназначены для размещения только для использования студентами кафедры «Компьютерные системы и технологии» НИЯУ МИФИ дневного отделения, изучающими курс «Программирование (Алгоритмы и структуры данных)».

Публикация (размещение) данных материалов полностью или частично в электронном или печатном виде в любых других открытых или закрытых изданиях (ресурсах), а также использование их для целей, не связанных с учебным процессом в рамках курса «Программирование (Алгоритмы и структуры данных)» кафедры «КСиТ» НИЯУ МИФИ, без письменного разрешения автора запрещена.

С3. Стандартная библиотека

Стандартная библиотека

- Работа с памятью
 - выделение и освобождение памяти
`malloc.h` `stdlib.h`
 - работа с памятью `memory.h` `string.h`
- Работа с текстовыми строками
 - обработка строк `string.h`
 - ввод `stdio.h`

Выделение и освобождение памяти

```
void * malloc(size_t n)
```

```
void * calloc(size_t n, size_t s)
```

```
void * realloc(void *ptr, size_t s)
```

```
void free(void *ptr)
```

Требуется явное преобразование типа:

```
char * pc = (char *) malloc(80);
```

```
int * pi = (int *)calloc(20, sizeof(int));
```


Обработка строк

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

```
char *strncat(char *s1, const char *s2, size_t n)
```

```
const char *strchr(const char *s, int c)
```

```
const char *strrchr(const char *s, int c)
```

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

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

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

```
char *strncpy(char *s1, const char *s2, size_t n)
```

Обработка строк

```
size_t strcspn(const char *s1, const char *s2)
```

```
size_t strspn(const char *s1, const char *s2)
```

```
size_t strlen(const char *s)
```

```
const char *strpbrk(const char *s1, const char *s2)
```

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

```
char *strtok(char *s1, const char *s2)
```

Ввод строки

`gets(char *);`

не контролирует размер памяти

`scanf("%Ls", buf)` (размер памяти – $L + 1$);

не вводит пробелы и символ `'\n'`

`scanf("%L[^\n]", buf);`

вводит любые символы, кроме `'\n'`

`'\n'` остается во входном потоке

Ввод строки

```
char buf[L + 1];
```

```
int n;
```

```
n = scanf("%L[^\n]*%*c", buf);
```

позволяет удалить из входного потока
символ '\n'

Варианты ввода данных

```
n = scanf("%L[^\n]", buf);
```

1. Вводится пустая строка

входной поток:

\n	---
----	-----

Результат:

$n = 0$

buf не меняет своего содержимого

входной поток:

\n	---
----	-----

Варианты ввода данных


```
n = scanf("%L[^\n]", buf);
```

2. Вводится строка длиной $k \leq L$

ВХОДНОЙ ПОТОК:

c	c	...	\n		
---	---	-----	----	--	--


Результат:



$n = 1$

buf:

c	c	...	\0
---	---	-----	----



ВХОДНОЙ ПОТОК:

\n		
----	--	--

Варианты ввода данных

```
n = scanf("%L[^\n]", buf);
```

3. Вводится строка длиной $k > L$

ВХОДНОЙ ПОТОК:

с	с	...	х	у	...	\n		
---	---	-----	---	---	-----	----	--	--

Результат:

$n = 1$

buf:

с	с	...	\0
---	---	-----	----

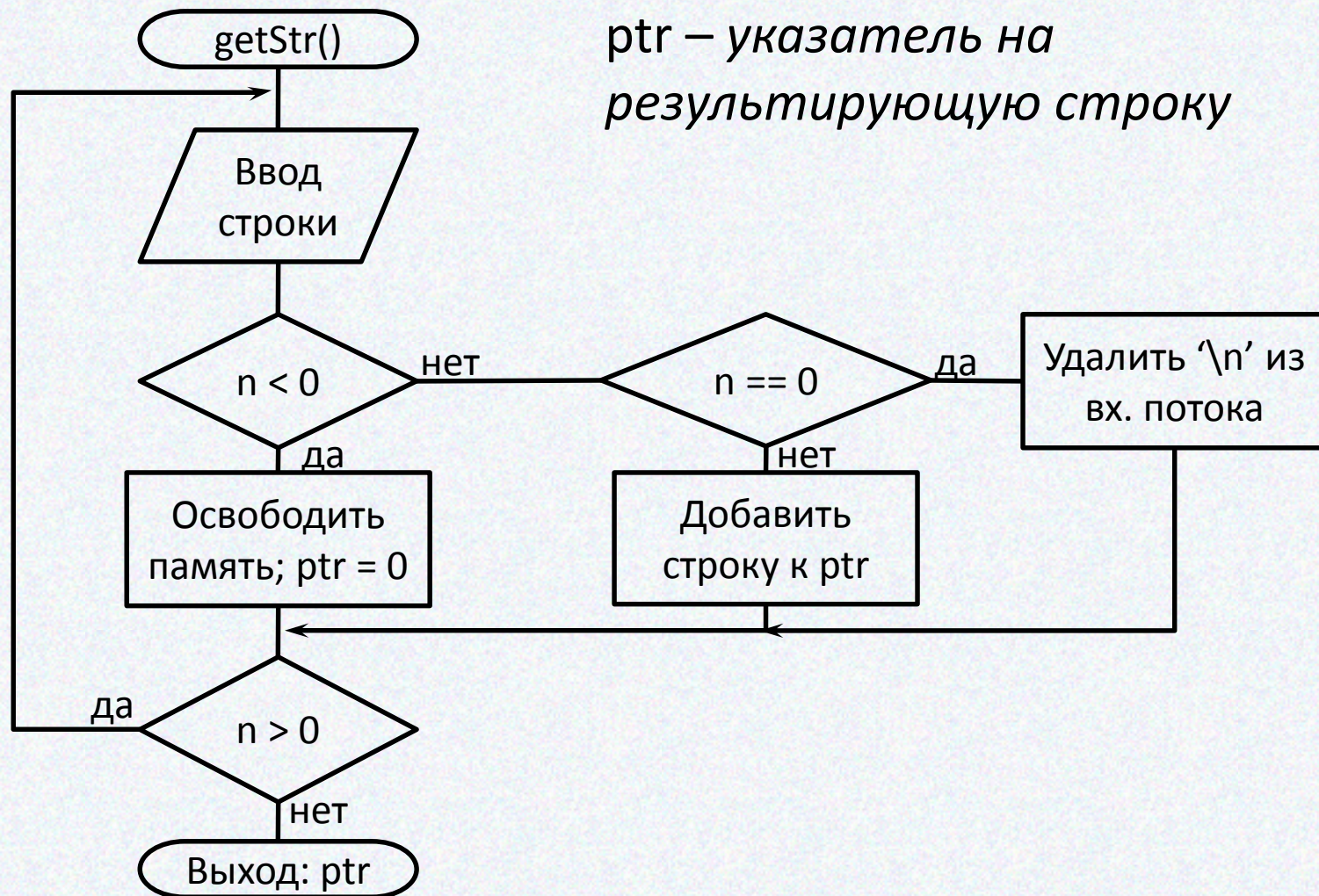
L

ВХОДНОЙ ПОТОК:

х	у	...	\n		
---	---	-----	----	--	--

Алгоритм ввода

*ptr – указатель на
результатирующую строку*



Алгоритм ввода

цикл {

 ввести строку: $n = \text{scanf}(\dots)$;

 анализ n :

$n < 0$ (конец файла или ошибка ввода):

 освободить память, результат = NULL

$n == 0$ (во входном потоке только '\n'):

 удалить из входного потока '\n'

$n > 0$:

 сформировать результирующую строку

} пока $n > 0$

Формирование строки

`ptr` – указатель на результирующую строку

`len` – длина результирующей строки

Исходное состояние: `ptr` – пустая строка, `len = 0`

`buf` – введенная строка

Вычислить `curLen` = длина строки в `buf`

Новая длина строки `ptr`: `len = len + curLen`

Перераспределить память для строки `ptr`

Добавить строку из `buf` к строке `ptr`

Реализация алгоритма

```
char *getstr()
{
    char *ptr = (char *)malloc(1);
    char buf[81];
    int n, len = 0;
    *ptr = '\0';
    do{
        n = scanf("%80[^\n]", buf); // n = scanf_s ("%80[^\n]", buf, 81);
        if(n < 0){
            free(ptr);
            ptr = NULL;
            continue;
        }
    }
```

Реализация алгоритма

```
if(n == 0)
    scanf("%*c");
else {
    len += strlen(buf);
    ptr = (char *) realloc(ptr, len + 1);
    strcat(ptr, buf);
}
} while(n > 0);
return ptr;
}
```