ПЗ-01: **Работа со строками**

Библиотека **string.h** предоставляет функции для работы со строками (*zero-terminated strings*) в Си, а также несколько функций для работы с массивами, которые сильно упрощают жизнь программистов.

***1. КОПИРОВАНИЕ ============================================================================***

**void \* memcpy (void \* destination, const void \* source, size\_t num);**

*Работа со строками - задача более глобальная, чем можно себе представить.*

*Так или иначе, практически каждое приложение связано с обработкой текста.*

|  |  |
| --- | --- |
| 1) Копирует участок памяти из **source** в **destination**, размером **num** байт.  Функция очень полезная, с помощью неё, например, можно скопировать объект или перенести участок массива, вместо поэлементного копирования.  Функция производит бинарное копирование, тип данных не важен.  Например, удалим элемент из массива и сдвинем остаток массива влево: | #include <conio.h>  #include <stdio.h>  #include <string.h>  #define SIZE 10  int main()  {  int a[SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  unsigned index;  int i;  for (i = 0; i < SIZE; i++) {  printf("%d ", a[i]);  }  printf("Enter index [0 .. 9]");  scanf("%ud", &index);  index = index < SIZE? index: SIZE-1;  memcpy(&a[index], &a[index+1], sizeof(int) \* (SIZE - index - 1));  for (i = 0; i < SIZE; i++) {  printf("%d ", a[i]);  }  getch();  } |

|  |  |
| --- | --- |
| 2) Функция меняет местами две переменные:    Здесь хотелось бы отметить, что функция выделяет память под временную переменную.  Это дорогостоящая операция.  3) Для улучшения производительности стоит передавать функции временную переменную, которая будет создана один раз:  #include <conio.h>  #include <stdio.h>  #include <string.h>  #include <stdlib.h> | #include <conio.h>  #include <stdio.h>  #include <string.h>  #include <stdlib.h>  void swap(void\* a, void\* b, size\_t size)  {  void \*tmp = malloc(size);  memcpy(tmp, a, size);  memcpy(a, b, size);  memcpy(b, tmp, size);  free(tmp);  }  int main()  {  float a = 300.456;  float b = 0.645;  printf("a = %.3f\nb = %.3f\n", a, b);  swap(&a, &b, sizeof(float));  printf("\n");  printf("a = %.3f\nb = %.3f", a, b);  getch();  } |

void swap(void\* a, void\* b, void\* tmp, size\_t size)

{

memcpy(tmp, a, size);

memcpy(a, b, size);

memcpy(b, tmp, size);

#include <stdio.h>

#include <string.h>

#include <conio.h>

void main ()

{

char str[] = "memmove can be very useful......";

puts(str);

memmove (str + 20, str + 15, 11);

puts(str);

getch();

}

}

int main()

{

float a = 300.456;

float b = 0.645;

float tmp;

printf("a = %.3f\nb = %.3f\n", a, b);

swap(&a, &b, &tmp, sizeof(float));

printf("\na = %.3f\nb = %.3f", a, b);

getch();

}

**void\* memmove (void \* destination, const void \* source, size\_t num);**

|  |  |
| --- | --- |
| Копирует блок памяти из **source** в **destination** размером **num** байт с той разницей, что области могут пересекаться.  Во время копирования используется промежуточный буфер, который предотвращает перекрытие областей. |  |

**char\* strcpy (char \* destination, const char\* source );**

Копирует одну строку в другую, вместе с нулевым символом. Также возвращает указатель на destination.

***Вариант 1 Вариант 2***

|  |  |  |
| --- | --- | --- |
| #include <stdio.h>  #include <string.h>  #include <conio.h>  #include <stdlib.h>  void main () |  | #include <stdio.h>  #include <string.h>  #include <conio.h>  #include <stdlib.h>  void main ()  {  char buffer[128];  char \*word = NULL;  char \*other = NULL;  scanf("%127s", buffer);  word = (char\*) malloc(strlen(buffer)+1);  other = strcpy(word, buffer);  printf("%s", other);  free(other);  getch();  } |
| {  char buffer[128];  char \*word = NULL;  scanf("%127s", buffer);  word = (char\*) malloc(strlen(buffer)+1);  strcpy(word, buffer);  printf("%s", word);  free(word);  getch();  } | |

**char\* strncpy (char\* destination, const char\* source, size\_t num);**

|  |  |
| --- | --- |
| Копирует только **num** первых букв строки.  0 в конец не добавляется автоматически.  При копировании из строки в эту же строку части не должны пересекаться (при пересечении используйте memmove) | #include <stdio.h>  #include <string.h>  #include <conio.h>  #include <stdlib.h>  void main ()  {  char word[] = "Aloha, Hawaii";  char aloha[20];  char hawaii[20];  printf(" %s", word);  strncpy(aloha, word, 5);  aloha[5] = 0;  strncpy(hawaii, &word[7], 7);  hawaii[7] = 0;  printf("\n %s, %s", aloha, hawaii);  getch();  } |

***2. Конкатенация строк =======================================================================***

|  |  |
| --- | --- |
| **char\* strcat (char \* destination, const char \* source);**  Добавляет в конец **destination** строку **source**, при этом затирая первым символом нулевой. Возвращает указатель на **destination**.  **char\* strncat (char \* destination, const char \* source, size\_t num);**  Добавляет в конец строки **destination** количество **num** символов второй строки. В конец добавляется нулевой символ. | #include <stdio.h>  #include <string.h>  #include <conio.h>  void main ()  {  char a[80];  char b[80];  scanf("%s", a);  scanf("%s", b);  strncat(a, b, strlen(b)/2);  printf("%s", a);  getch();  } |

***3. Сравнение строк ==========================================================================***

|  |  |
| --- | --- |
| **int strcmp (const char \* str1, const char \* str2);**  Возвращает 0, если строки равны, больше нуля, если первая строка больше, меньше нуля, если первая строка меньше.  Сравнение строк происходит посимвольно, сравниваются численные значения.  Для сравнения строк на определённом языке используется strcoll:  **int strcoll (const char \* str1, const char \* str2);**  Сравнение строк по первым **num** символам:  **int strncmp (const char \* str1, const char \* str2, size\_t num);**  Функция **qsort** библиотеки **stdlib.h** выполняет сортировку **num** элементов массива, на который ссылается указатель **first**.  Для каждого элемента массива устанавливается размер в байтах, который передается через параметр size. Последний параметр функции **qsort** — указатель **comparator** на функцию сравнения, которая используется для определения порядка следования элементов в отсортированном массиве.  Алгоритм сортировки используемый этой функцией сравнивает пары значений, путем вызова указанной функции сравнения, с двумя указателями на элементы массива. | /\* сортировка массива строк по первым трём символам \*/  #include <stdio.h>  #include <conio.h>  #include <string.h>  #include <stdlib.h>  int cmp(const void \*a, const void \*b)  {  return strncmp((char\*) a, (char\*) b, 3);  }  void main()  { int i;  char words[5][79] =  { "Solar",  "Obscurus",  "Tempestus",  "Ultima",  "Pacificus"  };  for (i = 0; i < 5; i++)  { printf("%s\n", words[i]);  }  qsort(words, 5, 80, cmp);  printf("\n");  for (i = 0; i < 5; i++)  { printf("%s\n", words[i]);  }  getch();  } |

**void qsort ( void \* first, size\_t number, size\_t size, int ( \* comparator ) ( const void \*, const void \* ) );**

Эта функция не возвращает никакого значения, но изменяет содержимое массива, на который указывает **first**. Таким образом, элементы массива занимают новые места, согласно отсортированному порядку.

***4. Поиск ====================================================================================***

|  |  |
| --- | --- |
| **1)  void\* memchr (void \* ptr, int value, size\_t num);**  Проводит поиск среди первых **num** байтов участка памяти, на который ссылается **ptr**, первого вхождения значения **value**, которое трактуется как **unsigned** **char**.  Возвращает указатель на найденный элемент, либо **NULL**. | #include <stdio.h>  #include <conio.h>  #include <string.h>  #include <stdlib.h>  void main()  {  char str[] = "Hello World!";  char \*ptr = NULL;  printf("%s\n", str);  ptr = (char\*) memchr(str, '\0', 4000);  if (ptr != NULL)  {  printf("first zero byte address is %p, strlen = %d", ptr, ptr - str);  } else  {  printf("no null byte in memory block");  }  getch();  } |

**2) char\* strchr (char \* str, int character);**

Возвращает указатель на место первого вхождения **character** в строку **str**.

Очень похожа на функцию **memchr**, но работает со строками, а не с произвольным блоком памяти.

|  |  |
| --- | --- |
| **3) size\_t strcspn ( const char \* str1, const char \* str2 );**  Возвращает адрес первого вхождения любой буквы из строки **str2** в строке **str1**.  Если ни одно включение не найдено, то возвратит длину строки.  **Пример** справа – найдём положение всех гласных в строке:    Здесь обратите внимание на строку **i++** после **printf**.  Если бы её не было, то **strcspn** возвращал бы всегда **0**, потому что в начале строки стояла бы гласная, и произошло зацикливание. | #include <stdio.h>  #include <conio.h>  #include <string.h>  void main()  { char str[] = "So if you want to love me\n"  "Then darling don't refrain\n"  "Or I'll just end up walking\n"  "In the cold November rain\n";  char vowels[] = "aeiouy";  int i;  i = 0;  while (str[i])  { i = i + strcspn(&str[i], vowels);  printf("%d ", i);  i++;  }  getch();  } |

Для решения этой задачи гораздо лучше подошла **функция**, которая возвращает указатель на первую гласную:

|  |  |
| --- | --- |
| **4) char\* strpbrk (char \* str1, const char \* str2)**  Функция очень похожа на **strcspn**, только возвращает указатель на первый символ из строки **str1**, который есть в строке **str2**.  Выведем все гласные в строке (см. код справа): | #include <stdio.h>  #include <conio.h>  #include <string.h>  void main()  {  char str[] = "Cos' it's a bittersweet symphony this life...\n"  "Trying to make ends meet, you're a slave to the money then you die.";  char vowels[] = "aeiouy";  char \*p = NULL;  p = strpbrk(str, vowels);  while (p)  { printf("%c ", \*p);  p++;  p = strpbrk(p, vowels);  }  getch();  } |

**5) char\* strrchr (char \* str, int character );**

Возвращает указатель на последнее вхождение символа в строку.

|  |  |
| --- | --- |
| **6) size\_t strspn (const char \* str1, const char \* str2);**  Возвращает длину куска строки **str1**, начиная от начала, который состоит только из букв строки **str2**.  **Пример**: вывести число, которое встречается в строке (код справа).  **7) char\* strstr (char \* str1, const char \* str2);**  Возвращает указатель на первое вхождение строки **str2** в строку **str1** (см. код ниже):  #include <stdio.h>  #include <conio.h>  #include <string.h>  void main()  {  char str[] = "I'll drown my beliefs\n"  "To have you be in peace\n"  "I'll dress like your niece\n"  "And wash your swollen feet\n";  char niece[] = "niece";  char\* p = strstr(str, niece);  printf("%s", p);  getch();  } | #include <stdio.h>  #include <conio.h>  #include <string.h>  #include <stdlib.h>  void main()  {  char str[] = "on 21st of May";  char nums[] = "0123456789";  char number[10];  /\* uintptr\_t - целочисленный тип без знака, который может хранить указатель данных \*/  uintptr\_t i;  //Определяем, где начинаются цифры  /\* size\_t - используется для индексации массива и подсчета циклов \*/  // целочисленный тип данных без знака  size\_t start = strcspn(str, nums);  /\* Определяем, где они заканчиваются, относительно start \*/  size\_t end = strspn(&str[start], nums);  for (i = 0; i < end; i++)  {  printf("%c", str[start+i]);  }  getch();  } |

**8) char\* strtok (char \* str, const char \* delimiters);**

|  |  |
| --- | --- |
| Разбивает строку на токены.  В данном случае токенами считаются последовательности символов, разделённых символами, входящими в группу разделителей. | #include <stdio.h>  #include <conio.h>  #include <string.h>  void main()  {  char str[] = "After working in India during the late 1970s and 1980s, "  "Shankar's profile in the West began to rise again in the mid-1990s "  "as his music found its way into club DJ sets, particularly in London.";  char delim[] = " \t\n\,.-";  char \*p = strtok(str, delim);  while (p != NULL)  {  printf ("%s\n",p);  p = strtok (NULL, delim);  }  getch();  } |

***5. + ещё функции ============================================================================***

**void \* memset (void \* ptr, int value, size\_t num);**

**memset** заполняет блок памяти символами value числом num штук. Например, можно заполнить массив или структуру нулями.

**size\_t strlen ( const char \* str );** // Самая популярная функция

**strlen** возвращает длину строки – число символов от начала до первого вхождения нулевого.

**int atoi (const char \* str);** // Конверсия строка-число

**atoi** переводит строку в целое

**double atof (const char\* str);** // Переводит строку в число типа double.

**long int atol ( const char \* str );** // Переводит строку в число типа long

Все функции такого рода имеют название XtoY, где X и Y - сокращения типов.

A обозначает ASCII. Соответственно, имеется обратная функция itoa (больше нет:)).

Таких функций в библиотеке stdlib.h очень много, все их рассматривать не хватит времени ПЗ-02.

**Форматированный ввод и вывод в буфер**

Можно также выделить две функции **sprintf** и **sscanf**. Они отличаются от **printf** и **scanf** тем, что выводят данные и считывают их из буфера. Это, например, позволяет переводить строку в число и число в строку.

int i;

char buffer[128];

sprintf(buffer, "%d", i);

printf("%s\n", buffer);