АЛГОхитрости

Типовые алгоритмические приёмы работы с символами и строками

1. Вывод на экран (печать) символа и его кода (номера в таблице ASCII).
2. Организация терминального условия в цикле до конца строки.
3. Символьные рамочные терминальные условия цикла.
4. Строка – целое число (в цикле).
5. Строка – вещественное число.
6. Удаление начальных пробелов в строке.
7. Особенности ввода строк оператором **scanf** со спецификатором формата ввода **%s**.

C / C++

АЛГОхитрости

Типовые алгоритмические приёмы работы с символами и строками

1. **Вывод на экран (печать) изображения символа ASCII и его кода (порядкового номера в таблице).**

Используются спецификаторы формата вывода:

Спецификатор формата %c выведет на экран собственно символ,
Спецификатор формата %i (%d) выведет на экран код символа.

Пример:

```
printf("Символ: %c Код: %i\n", F, F);
```

Примечание:
Операция `int n = 'h'+'e'+'l'+'l'+'o';`
выполнит арифметическое суммирова-ние значений кода символов: n = 532

АЛГОхитрости

C / C++

Типовые алгоритмические приёмы работы с символами и строками

2. Организация терминального условия в цикле до конца строки.

Терминальное условие автоматически становится ложным, когда встречает при пробеге по строке 0-символ (\x0 – 16-иричный ноль).

Пример:

```
while (str[i]); // цикл продолжается до \x0
```

Но можно использовать функцию:

strlen (str) – текущая длина строки str,

Пример:

```
while (i <= strlen(str));
```

АЛГОхитрости

Типовые алгоритмические приёмы работы с символами и строками

3. Символьные рамочные терминальные условия цикла.

В качестве терминального условия в циклах и рекурсивных функциях обработки строк может использоваться рамочный диапазон символов:

Например,
для проверки попадания символа строки в диапазон символов-цифр терминальное условие при пробеге по строке будет:

```
while ((st[i] >= '0') && (st[i] <= '9'))
```

– в C \ C++

Сравните в Pascal

```
while (st[i]>='0') and (st[i]<='9') and (i<=Length(st))
```

C / C++

АЛГОхитрости

Типовые алгоритмические приёмы работы с символами и строками

4. Строка – целое число (в цикле).

```
#include <stdio.h>
#include <conio.h>
void main()
// Проверка, является ли строка целым числом
{
    char st[20];    // строка
     int i;             // номер проверяемого символа строки
  printf("\n Введите число и нажмите Enter \n");
  printf ("-> ");
  scanf ("%s", st);
    i = 0;
    while ((st[i] >= '0') && (st[i] <= '9'))
      i++;
printf("/n Введённая строка ");
   if (st[i])     // st[i] -  \0, если введены только цифры
     printf("не ");
     printf("является целым числом.");
   printf("\n\nДля завершения нажмите Enter");
   getch();
}
```

C / C++

АЛГОхитрости

Типовые алгоритмические приёмы работы с символами и строками

5. Строка – вещественное число без знака.

```
#include <stdio.h>
#include <conio.h>
void main()
// Является ли строка дробным числом без знака
{  char st[20];   // строка
   int i;         // номер проверяемого символа строки
   int ok=0;   // пусть, строка - не дробное число
printf("\n Введите число и нажмите Enter\n");
  printf ("-> ");    scanf ("%s", st);
 i = 0;
if((st[i]>='0')&&(st[i]<='9')) //первый символ - цифра
   {  while ((st[i] >= '0') && (st[i] <= '9'))
         i++;
```

см. продолжение

продолжение

```
 if (st[i] == '.')  // за цифрами должна быть точка
        {  i++;
    // за точкой должна быть хотя бы одна цифра
      if ((st[i] >= '0') && (st[i] <= '9'))
        {   // и ещё цифры
          while ((st[i]>='0')&&(st[i]<='9'))
              i++;
            ok = 1;  // похоже строка - дробное число
        } } }
 printf("\nСтрока %s ", st);
   if (st[i] || !ok)
    printf("не ");
    printf("является дробным числом без знака.");
printf("\n\nДля завершения нажмите Enter");
  getch();   }
```

C / C++

АЛГОхитрости

Типовые алгоритмические приёмы работы с символами и строками

6. Удаление начальных пробелов в строке.

```
#include<stdio.h>
#include<string.h>
main()
{  int i;  char str[100]; // Массив для строки
printf("\nВведите строку символов:\n");
   gets(str);  // Вводим строку
   while (str[0]==' ') /*если первый символ пробел, то сдвигаем массив влево  на одну позицию */
   {    for (i=0; i<strlen(str); i++)
str[i]=str[i+1];       }
printf("\nИсправленная строка:\n%s\n",str);
}
```

Более оптимально сделать так:

```
int k=0;
while (str[k]==' ')    k++;   // считаем пробелы
{      // смещаем строку  на k символов
    for (i=0; i<=strlen(str); i++)
        str[i]=str[i+k];
}
```

АЛГОхитрости

Типовые алгоритмические приёмы работы с символами и строками

7. Особенности ввода строк функцией scanf со спецификатором формата ввода %s.

Для чтения из входного потока строки можно использовать функцию scanf() со спецификатором преобразования %s. Использование спецификатора преобразования %s заставляет scanf() читать символы из буфера клавиатуры до тех пор, пока не встретится какой-либо разделитель. Читаемые символы помещаются в элементы символьного массива, на который указывает соответствующий аргумент, а после введенных символов еще добавляется символ конца строки ('\x0').

Разделителем может быть пробел, разделитель строк, табуляция, вертикальная табуляция или подача страницы. В отличие от функции gets(), которая читает строку, пока не будет нажата клавиша <ENTER>, scanf() читает строку до тех пор, пока не встретится первый разделитель. Это означает, что scanf() нельзя использовать для чтения строки "это испытание", потому что после пробела процесс чтения прекратится.

При вводе строки функцией scanf разделители, расположенные перед первым значащим символом строки – не вводятся (пропускаются).

Пример: ввод строки " привет всем":

```
#include <stdio.h>
int main(void)
{ char str[80];
printf("Введите строку: ");
scanf("%s", str);
printf("Вот Ваша строка: %s", str);
return 0; }
```

Программа выведет только часть строки, то есть слово "привет", без пробелов.