Лабораторная работа №7

Строки в C/C++

1. Цель работы:

1) Получение практических навыков при работе со строками
2) Получение практических навыков при передаче строк в функции.

2. Краткие теоретические сведения

2.1. Текстовые данные

В языке С текстовая информация представляется двумя типами данных: с помощью символов и строк - массивов символов.

Символьный тип данных

Значением данных символьного типа является любой символ из набора всех символов компьютера или его код. Каждому символу соответствует порядковый номер (код) в диапазоне 0..255. Для кодировки символов первой половины диапазона используется код ASCII или более современные стандарты в последних версиях языка Си.

При написании программ символьные данные могут быть представлены либо константами, либо переменными.

Символьная константа представляет собой одиночный символ, заключенный в апострофы, например:

‘Y’ ‘!’ ‘_’ ‘Д’

Символьная переменная объявляется с помощью ключевого слова char, например:

char cr;

Во внутренней памяти компьютера каждый символ занимаем 1 байт.

2.2 Ввод-вывод символьных данных

Для ввода символьных данных используются функции: scanf() – форматированный ввод, getchar() или getch() – специальные функции для ввода символа. Для форматного ввода и вывода символьных констант используется спецификатор (формат) %c. Необходимо помнить, что нажатие любой небуквенной клавиши при вводе ([пробел], [Enter] и др.) будет значимым и восприниматься как символ.

Пример 1. Организовать ввод символьных переменных:

```
a='i'   b='j'   c='k'
      main()
      {     char a,b,c;
           printf("Введите исходные данные");
            scanf("%c%c%c",&a,&b,&sc);
              . . .
     }
```

При вводе символы набираются без апострофов и пробелов:

ijk [Enter]

Символ клавиши [Enter] выходит за пределы списка ввода, поэтому он игнорируется.

При вводе символьной информации с помощью функции getchar() надо помнить, что функция переводит программу в состояние ожидания, но при нажатии клавиши

символ выводится на экран. А, например, при выполнении следующего фрагмента программы

```
printf("Введите исходные данные");
a=getch();b=getch();c=getch();
```

переменные будут введены, но на экране их значения не отразятся.

Для вывода символьных данных используются функции printf() и putchar().

Пример 2. Организовать вывод указанных выше переменных на экран в одну строку. Запись оператора вывода будет следующей:

```
printf("%c%c%c\n",a,b,c);
```

На экране будет отображено:

```
ijk
```

Если использовать для вывода функцию putchar():

```
putchar(a); putchar(b); putchar(c);
```

на экране будет отображен тот же результат.

2.3 Обработка символьных данных

Поскольку символы в языке С++ упорядочены, к ним можно применять операции отношения (>, >=, <, <=, = =, !=). Это дает возможность записывать логические выражения с символьными данными в условных операторах. Например, оператор

```
if (ch =='!' ) ch ='.';
```

сравнивает значение переменной ch с символом '!' и в случае их равенства следующая команда заменяет в символьной переменной ch восклицательный знак точкой.

Символьные данные могут использоваться и в операторах цикла for. Так, при выполнении операторов:

```
for( ch='a'; ch>='d'; ch++)  printf("%c",ch);
```

в строку экрана выводится последовательность:

```
a b c d
```

Если значение символьной переменной вывести с помощью спецификатора для целых чисел %d, то на экране отобразится код символа. Например:

```
for(ch='a'; ch>='d'; ch++)  printf("%d ",ch);
```

на экран будет выведено:

```
97 98 99 100
```

Над символьными данными можно выполнять арифметические операции сложения и вычитания. Так, например, операция ch++; из предыдущего примера увеличивает код символа, хранящегося в переменной ch на 1. Или, выполняя операцию ch='a'-'A'; будет получена разница кода большой (А) и маленькой буквы (а) латинского алфавита. Так, например, если в символьной переменной ch1 хранится маленькая буква алфавита, то, выполнив действия:

```
char ch,ch1,ch2;
ch='a'-'A';
ch1='k';
ch2=ch1-ch;
printf("%c-%d %c-%d\n",ch1,ch1,ch2,ch2);
```

в переменную ch2 запишется та же буква, только большая, а на экран будет выведено

```
k-107  K-75
```

2.4 Строки

Строка в С++ – это массив символов, заканчивающийся нуль-символом – '\0' (нуль-терминатором). По положению нуль-терминатора определяется фактическая длина строки. Количество элементов в таком массиве на 1 больше, чем изображение строки.

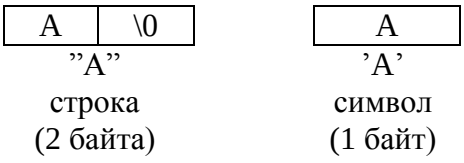

Рис. 1. Представление строки и символа

```
char s[80] = "Язык программирования Си";
```

Символы в кавычках будут записаны в начало массива ***s***, а затем - признак окончания строки ***'\0'***.

При описании строки можно также написать так:

```
char s[] = "Язык программирования Си";
```

В этом случае компилятор подсчитает символы в кавычках, выделит памяти на 1 байт больше и занесет в эту область саму строку и завершающий ноль.

Присвоить значение строке с помощью оператора присваивания нельзя. Поместить строку в массив можно либо при вводе, либо с помощью инициализации.

```
char s1[10]="string1";//инициализация
char s2[]="string2";//инициализация
char s3[10];
cin>>s3;//ввод
//выделение памяти под динамическую строку
char *s4=new char[strlen(s3)+1];
strcpy(s4,s3);//копирование строки s3 в строку s4
```

Ввод-вывод строковых данных

При вводе строк, как и символов, используется функция scanf(). При этом для форматного ввода и вывода строк используется спецификатор %s. Однако нажатие клавиши [Enter] или клавиши [пробел] не является значимым символом. При вводе строки с помощью функции scanf() нажатие одной из этих клавиш формирует символ конца строки. Таким образом надо помнить, что функция scanf() позволяет записать в строку только одно слово.

Пример. Организовать ввод ФИО студента.

```
char fam[20];
printf ("Введите фамилию и инициалы студента");
 scanf("%s", fam);
```

На клавиатуре строка набирается без кавычек, например:

Андреева С.В. [Enter]

Одновременно с вводом строки в байт с индексом восемь запишется символ с кодом 0. Инициалы студента в эту строку записаны не будут, так как пробел после фамилии будет воспринят командой scanf как конец строки.

Для ввода текста содержащего пробелы следует использовать специальную функцию gets(). При вводе строки с помощью этой функции только нажатие клавиши [Enter] сформирует символ конца строки.

Так, например, в предыдущей задаче:

```
char fam[20];
printf(“Введите фамилию и инициалы студента”);
gets(fam);
```

использование функции gets() позволит записать в строку fam не только фамилию, но и инициалы.

Вывод строк осуществляется с помощью функции printf() и специальной функции puts(). Например, оператор

printf("| %20s|",fam);

выведет на экран в правую часть поля из 20 позиций строку fam:

| Андреева С.В.|

Специальная функция puts() позволяет вывести содержимое строки и переводит курсор на следующую строку. Например:

putchar('|'); puts(fam); putchar('|');

приведет к получению следующего результата:

|Андреева С.В.
|

Последний символ будет выводиться в следующей строке экрана.

Обработка строковых данных

К любому символу строки можно обратиться как к элементу одномерного массива, например, запись st[i] определяет i-ый символ в строке st. Поэтому при решении некоторых задач обработку строковых данных можно проводить посимвольно, организуя циклы для просмотра строки.

Например: Дано предложение. Определите количество слов в нем.

Решение:

Слова в предложении разделяются пробелами. Подсчитав количество пробелов, можно определить количество слов, учитывая, что между словами введен только один пробел.

```
#include "stdafx.h"
#include<string.h>
int main()
{ char slova[120];
    int i, n, k=1;
    printf("Введите предложение\n");
    gets(slova);
    n= strlen(slova); // функция strlen() возвращает длину строки
    for(i=0;i<n; i++)
         if(slova[i]==' ')k++; //поиск и подсчет пробела
     printf("k=%d\n",k);
      return 0;
}
```

2.5 Стандартные функции обработки строк

Большинство действий над строками реализуется с помощью стандартных функций. Библиотека языка Си содержит большое количество таких функций, прототипы которых определяются в заголовочном файле string.h. Рассмотрим некоторые из них.

Сравнение строк:

strcmp(str1,str2) – сравнивает две строки str1 и str2 и возвращает 0, если они одинаковы; результат отрицателен, если str1<str2 и положителен, если str1>str2.

strncmp(str1, str2, kol) – сравниваются части строк str1 и str2 из kol символов. Результат равен 0, если они одинаковы.

Сравнение двух строк выполняется последовательно слева неправо с учетом кодировки символов. Например, сравнивая стоки st1 и st2

```
char  st1[10]="Пример";
```

```
char  st2[10]="ПРимер";
int  a;
 if  (strcmp(st1,st2)>0)
      a=1;
else
      a=2;
```

переменной a будет присвоено значение 1, так как код символа 'р' больше кода символа 'Р'.

Сцепление строк

strcat(str1,str2) - сцепление строк в порядке их перечисления.
strncat(str1,str2,kol) – приписывает kol символов строки str2 к строке str1.

Функция служит для объединения двух строк в одну. Например, в результате выполнения операторов:

```
char fam[] = "Андреева С.В. ";
char pr[7]= "       ";   //7 пробелов
strcat(fam ,pr);
 printf("|%20s|", fam);
```

на экран выведется строка:

|Андреева С.В. |

Заметим, что строка вывода занимает поле в 20 позиций, а переменная fam располагается в левой части поля.

Определение длины строки

strlen(str) – определяет длину строки str.

Пример. Определить длину строки

```
char fam[] = "Андреева С.В.";
printf("%d",strlen(fam));
```

функция strlen() вернёт значение равное 13 (символов).

Копирование строк

strcpy(str1,str2) – копирует строку str2 в строку str1.
strncpy(str1, str2, kol) – копирует kol символов строки str2 в строку str1.

Пример. Скопировать фамилию сотрудника в переменную fam и вывести на экран.

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

int main()
 { char fam[15];
    char *str = " Андреева С.В.";
    strcpy(fam, str);
    printf("|%s|\n", fam);
    return 0;
}
```

В результате выполнения данных операторов на экран будет выведена строка:

|Андреева С.В.|

Поиск символа в стоке

strchr(st, ch) - функция поиска адреса символа ch в строке st. Результатом выполнения поиска является адрес найденного символа в строке st, иначе возвращается

нулевой адрес. Чтобы вычислить порядковый номер символа ch в строке, можно из адреса P вычесть адрес начала строки.

Пример. В заданной фамилии определить порядковый номер символа ‘n’.

```
#include "stdafx.h"
#include<string.h>
int main()
{ char fam[] = "Ivanov";
   char faml[20];
   char  a='n';
   char *p;
    p=strchr(fam,a);
    if(p)
         printf("|%s|%d\n", fam, p-fam);
    else
     printf("нет такого символа в фамилии!\n");
     return 0;
}
```

2.6 Пример программы для задачи с текстовыми данными

Исходным текстом является предложение, заканчивающееся точкой. Слова в предложении отделяются друг от друга одним пробелом. Определить самое длинное слово в предложении.

```
#include "stdafx.h"
#include<string.h>
int main()
{ char slovo[12],x[120];      // описание строк
int i,m=0,n,k=0;
gets(x);                      // ввод строки x
for(i=0; i<strlen(x); i++)  //цикл до конца строки x
if(x[i]!=' ')   k++;          // считаем символы до пробела
else
    { if (k>m){ m=k;n=i;}   //поиск мах значения счетчика к
     k=0;
    }
     k=0;
     for(i=n-m;i<n;i++)     //выбор из строки x самого длинного слова
     slovo[k++]=x[i];
     slovo[k]=0;

     printf("%s    \n%s\n",slovo,x); /*вывод найденного слова и всей
                                       строки x  */
printf("%d  %d\n",strlen(slovo),strlen(x));//вывод их длин
return 0;
        }
```

2.7. Передача строк в качестве параметров функций

Строка в Си++ - это массив символов, заканчивающийся нуль-символом – ‘\0’ (нуль-терминатором). По положению нуль-терминатора определяется фактическая длина строки. Количество элементов в таком массиве на 1 больше, чем изображение строки.

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

Строки при передаче в функции могут передаваться как одномерные массивы типа char или как указатели типа char*. В отличие от обычных массивов в функции не указывается длина строки, т. к. в конце строки есть признак конца строки /0.

```
//Функция поиска заданного символа в строке
int  find(char *s,char c)
{
for (int I=0;I<strlen(s);I++)
if(s[I]==c) return I;
return -1
}
```

3. Постановка задачи

1. Ввести с клавиатуры строку символов и обработать ее в соответствии со своим вариантом, используя функции.

4. Варианты

Вариант	Строки
1	Удалить все гласные буквы из строки.
2	Подсчитать количество слов в строке.
3	Перевернуть каждое четное слово в строке.
4	Удалить каждое четное слово из строки.
5	Отсортировать слова в строке в лексикографическом порядке (по алфавиту).
6	Удалить из строки все слова, начинающиеся на гласную букву.
7	Удалить из строки все слова, заканчивающиеся на гласную букву.
8	Удалить все гласные буквы из строки.
9	Подсчитать количество слов в строке.
10	Перевернуть каждое четное слово в строке.
11	Удалить каждое четное слово из строки.
12	Отсортировать слова в строке в лексикографическом порядке (по алфавиту).
13	В исходном предложении перед каждым словом поставить знак ?.
14	В заданном предложении найти самое короткое и самое длинное слово.
15	Удалить все гласные буквы из строки.
16	Подсчитать количество слов в строке.
17	В исходном предложении удалить все символы пробела. Вывести преобразованный текст и количество удаленных пробелов.

5. Методические указания

1. Ввод/вывод строк организовать с помощью функций:

- char* gets(char*s)
- int puts(char *s)

2. Для обработки строк использовать стандартные функции из библиотечного файла <string.h>

3. Функция main() должна содержать только описание строк и вызовы функций для формирования, печати и обработки строк