ЛЕКЦИЯ 11
Текстовые данные

Символьный тип данных

Строковый тип данных

(уч. пособие со стр.119)

Символьный тип данных

1) **Объявление символьных переменных**
2) ***Ввод-вывод символьных данных***
3) ***Обработка символьных данных***

Объявление символьных переменных

- **Значением данных символьного типа является любой символ из набора всех символов клавиатуры компьютера.**
- При написании программ символьные данные могут быть представлены **либо константами, либо переменными.**
- **Символьная переменная** объявляется с помощью ключевого слова **char**, например:

char AB;

- Во внутренней памяти компьютера каждый символ занимаем 1 байт

Примеры объявления символьных данных

```
int _tmain(…)
 {
```

char m; //объявлена символьная переменная с именем m

char ch='!'; /*объявлена символьная переменная с именем ch и инициализирована значением символа **!** */

char r=' A'; /*объявлена символьная переменная с именем r и инициализирована значением символа **A** */

…

Ввод символьных данных

Для ввода символьных данных существует в языке Си функции:

scanf() – форматированный ввод,

или cin - потоковый ввод

и специальная функция

getch()

Для форматного ввода символьных значений функцией **scanf** используется спецификатор (формат) **%c.**

Пример 1.

Организовать ввод символьных значений переменных a='i'; b='j'; c='k'

с помощью **функции scanf**

```
int _tmain()
{   char a,b,c;        //описание переменных
        printf("Введите исходные данные");
        scanf("%c%c%c",&a, &b, &c);
        ……….}
```

- При вводе символы набираются без апострофов и пробелов:

ijk [Enter]

Организовать ввод символьных переменных
a='i'; b='j'; c='k'

- С помощью функции **cin**

```
 void main()
 {  char  a,b,c;
    printf("Введите исходные данные");
   cin>>a>>b>>c;
   …… }
```

- При вводе символы набираются без апострофов и пробелов:

ijk [Enter]

Организовать ввод символьных переменных
a='i'; b='j'; c='k'

- С помощью функции **getch()**

```
 void main()
{  char  a,b,c;
   printf("Введите исходные данные");
  a=getch();
  b=getch();
  c=getch();
  …… }
```

- При вводе символы набираются без апострофов:

ijk [Enter]

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

вывод символьных данных

Для вывода символьных значений переменных в языке Си существует три функции:

- **printf()**
- **cout**
- и специальная функция **putch()**

Пример вывода значений символьных переменных на экран

С помощью функции printf

```
main()
{    char a,b,c;
a='*';   b='m';  c='!';
 printf("значения данных\n");

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

С помощью функции cout

```
main()
{

char a='*',  b= 'm',   c='!';
 printf("значения данных\n");

cout<<a<<"   "<<b<<"   "<<c<<"\n"

……
}
```

На экране будет отображено:

значения данных

* m !

Обработка символьных данных

- Поскольку коды символов в языке Си/С++ упорядочены, к ним **можно применять операции отношения**

(>, >= , < , <= , == , !=).

```
int   _tmain()
{
    char   ch;
     ch=getch();
        if   (ch  = ='!' )   ch ='.';
```

Значение переменной ch сравнивается с символом **!** и в случае равенства значение **!** заменится **на точку**.

Обработка символьных данных

- Символьные данные могут **управлять работой оператора цикла for.**

- void main()
- { **char ch;**
- **for(ch='a'; ch<=‘f'; ch++)**
- printf("%c ",ch);
- …….

на зкране появятся все символы от a до f

a b c d e f

```
for(ch='a';   ch<=‘f’;    ch++)
        printf("%d ",    ch);
```

на экран будет выведено:

97 98 99 100 101

Обработка символьных данных

- Над символьными данными можно выполнять **арифметические операции сложения и вычитания**

```
void main()
{
    char ch,ch1,ch2;

    ch='a'-'A';  //  вычисление разницы кодов  букв  a и A

     ch1='k';
    ch2=ch1-ch; // код буквы k будет изменен  на код  буквы K

    printf("%c  -  %d \n“,  ch2,  ch2);
```

операции будут производиться над кодами символов, а на экран будет выведена следующая информация:

K - 75

Строковый тип данных

1) **Объявление строковых переменных**
2) ***Ввод-вывод строковых данных***
3) ***Обработка строковых данных***

Объявление строковых переменных

- Строковая константа - это строка, заключенная в кавычки, например:

  “Язык программирования C++!”

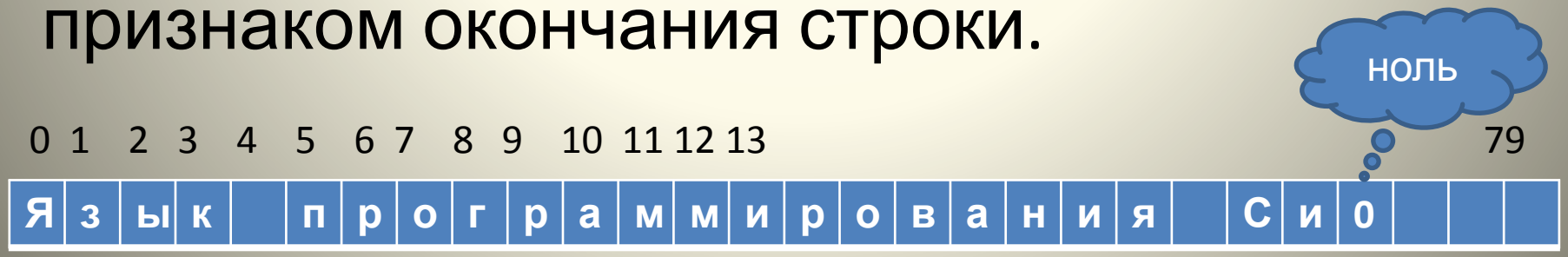
- Строковая переменная или строка представляет собой **массив символов**, поэтому и объявляется она именно так:

```
void main()
{       char st[30];
```

Примеры объявления строковых переменных

```
void main()
{
char s[80] = "Язык программирования Си";
char str[] = "Язык программирования Си";
……..
```

строка отличается от массива тем, что она заканчивается символом с кодом 0 - признаком окончания строки.



Ввод строковых данных

- Для ввода строк, как и символов, используется функция

scanf() со спецификатором (форматом) %s.

или

- специальная функция **gets().**

Пример. Организовать ввод ФИО студента

Функция scanf

```
void main()
{
char   fam[25];
printf("Введите фамилию
    студента");
scanf("%s",fam);
……
```

- Замечание: функция scanf читает символы до пробела

Функция gets

```
#include “string.h”
void main()
{
char    fam[25];
 printf(“Введите фамилию и
    инициалы студента”);
      gets(fam);
……..
```

- Замечание: для функции gets все символы значимые (пробел тоже)

Вывод строковой переменной

- Вывод строки осуществляется с помощью функции

printf()

или

специальной функции **puts().**

Пример вывода строки

функцией printf

```
void main()
{
  char fam[20]=“АНДРЕЕВА
     А.”
…….

  printf(“   %20s”,    fam);
……..
```

функцией puts().

```
void main()
{
  char fam[20]=“АНДРЕЕВА
     А.”
…….

 puts(fam);
……..
```

Обработка строковых данных
Стандартные функции и процедуры обработки строковых данных

подключать файл string.h

- **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.

Пример использования функций сравнения

```
#include<string.h>
void main()
{
 char  st1[10]="Пример";
  char  st2[10]="ПРимер";
    int  a,d;
       d= strcmp(st1,st2);
       if  (d!=0)
 {  cout<<"st1 не   равна  st2";  a=1;}
       else                a=2;
….}
```

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

Стандартные функции обработки строковых данных

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

Функции служат для объединения двух строк в одну.

Пример использования функций сцепления

```
#include<string.h>
void main()
{
char fam[] = "Андреева С.В. ";
char pr[20]= “ студентка гр. С-07 ";
strcat(fam ,pr);
 printf(“%40s”, fam);
```

- на экран будет выведена строка:
  Андреева С.В. студентка гр. С-07

Стандартные функции и процедуры обработки строковых данных

- **Определение длины строки**

strlen(str) – определяет длину строки str.

Пример. Определить длину строки

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

- функция strlen() вернёт значение равное 13 (количество символов в массиве fam).

Стандартные функции и процедуры обработки строковых данных

- **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<stdio.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.

Стандартные функции и процедуры обработки строковых данных

- **Поиск символа в строке**
- char *P=**strchr**(st, ch) - функция поиска адреса символа ch в заданной строке st.

Результатом выполнения поиска является адрес P 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 ‘a’.

```
#include<stdio.h>
#include<string.h>
    int main()
  {
    char fam[20];
    printf(“введите фамилию\n”);
     gets(fam);
       char   *p;
     p=strchr(fam, ‘a’);
      if   (  p!=0   )        printf("%s     %d\n",    fam,    p-fam);

       else      printf(“символа «a»  нет    в фамилии!\n");
   return 0;
   }
```

Вычисление номера

Примеры программирования задач с текстовыми данными

- 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**.

Пример 1: Дано предложение. Определите количество слов в нем.

```
#include<stdio.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++;
           //сравнение каждого символа предложения с пробелом

             printf("k=%d\n",k); //вывод значения k (счетчика пробелов)
     return 0;
   }
```

Пример программирования 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.

Программа (работа с charМАССИВОМ)

```
#include<stdio.h>
#include<string.h>//подключение библиотеки для
  РАБОТЫ С СИМВОЛЬНЫМИ ДАННЫМИ
int main()
{ char slovo[12],x[120];
// описание строковых переменных
  int i,max=0,n,k=0;
//описание счетчика, переменной для индекса, max
gets(x);  // ввод предложения в МАССИВ x
for(i=0; i<strlen(x)-1; i++)   // организация
  цикла от нулевого элемента до конца строки x
   { if(x[i]!=' ')k++;//считаем символы неравные
  пробелу
          else
          { if ( k>max ){ max=k;   n=i;}
  /*сохранение в max-длину и в n позицию, на
  которой закончилось очередное длинное слово в
  предложении */
```

Программа(продолжение)

```
  k=0;
 for(   i=n-max;   i<n;   i++)
slovo[k++]=x[i];  /* перезапись длинного слова из
                     предложения в  массив   slovo*/



printf("%s   \n",slovo);//выводим содержимое
  массива slovo

printf("%d %d\n",   strlen(slovo), strlen(x));
//выводим длину массива slovo и длину всего
  предложения  x
return 0;
}
```

Тип данных СТРОКИ (string)

- Для его использования необходимо подключить библиотеку **<string>**
- **Пример**

```
#include<string.h>
int main()

{  char c1[50],  c2[50],  c3[50];//строки с завершающим нулем
    strcpy(  c1,  “миру мир!”);

     strcpy(c2,c1); //копирование строки c1 в  c2
     strcat(c1,c2);
puts(c1);
}
```

Пример2

```
#include<string>
int main()

String  s1,  s2,  s3;
    s1=“ миру мир”;

    s2=s1; //копирование строки s1 в  s2
s3=s1+s2;      //объединение строк
puts(s3);
```

Функции работы со STRING

(приложение в уч. пособии)

```
#include <string>
#include <iostream>
using namespace std;
int main()
{  int n;
  String  s1(“прекрасная королева”);
 String s2(“ле”);
 String  s3( “корова”);
```

s3.insert(4,s2); /* вставка строки s2 в строку s3 после четвертого символа */

s1.erase(0,3); // удаление первых трех символов в строке s1

n= **s1.find(‘ o’);** // ищет первое вхождение символа ‘o’ в строку s1

Сравнение символьных массивов

```
if  (  strcmp(c1,c2)<0  )    cout  << c1;
else  cout   << c2;
```

Сравнение строк

- If (s2<s3)   cout << s2;
  else cout <<s3;