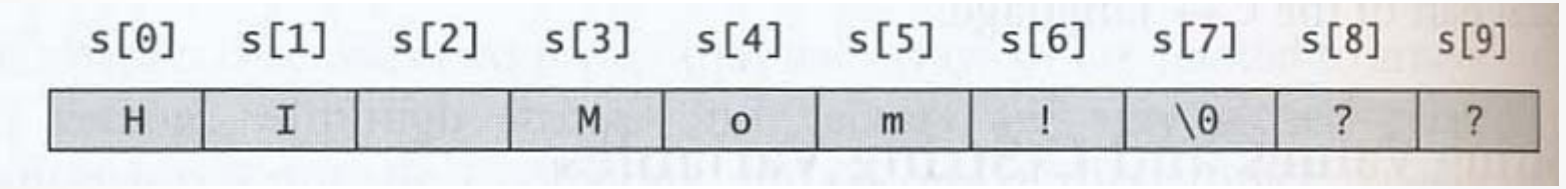
# Стрингове

Базово програмиране
Лекция 7
Милена Карова
кат. КНТ ТУ-Варна

# Използване на масиви

- Стрингове
- Вектори
- 2 типа вектори: представени чрез типа char и типа string
- C string е името на първия тип – стринг от символи представен чрез масив от символи с край ‘\0’
- class string  е модерният начин за използване на стрингове в C++.

# C-String променливи и стойности

- ‘\0’  нулев символ или терминатор на низа
- ’\0’ може да бъде символна променлива или индексирана променлива
- Декларация за 9 букви и нулев символ `char s[10];`
- char array_name[max size of string+1]
- 

  int стойността в скобите -  показва дължината на стринга. ‘\0’ определя докъде да се чете стринга
- Дефиниране на указател към стринга **char *s**;

# Разлика между стринг и масив от символи

- Разликата е в символа '\0'
- Променливата от тип C string е масив от символи, но използвани по друг начин.
- Инициализация на C- string променливи

```
char my_message[20] = "Hi there.";
```

```
char short_string[] = "abc";
```

```
char short_string[4] = "abc";
```

- Еквивалентни
- не са еквивалентни

```
char short_string[] = "abc";
char short_string[] = {'a', 'b', 'c'};
```

- Поелементна обработка

```
char our_string[5] = "Hi";
```

```
int index = 0;
while (our_string[index] != '\0')
{
    our_string[index] = 'X';
    index++;
}
```

# Присвояване на C-String стойност

```
char happy_string[7] = "DoBeDo";
happy_string[6] = 'Z';
```

- В този пример се премахва ‘\0’.
- Не можете да използвате оператор за присвояване = или знак за равенство== за работа със стрингове. Това са масиви и не могат да използват тези оператори.
- Присвояването е по-сложен процес

```
char a_string[10];
a_string = "Hello";
```

*Illegal!*

```
char happy_string[7] = "DoBeDo";
```

- Присвояване може да се осъществи само по време на деклариране. Това е инициализация, а не присвояване.

# Присвояване на C-String стойност

- Присвояване с помощта на функцията strcopy

```
strcpy(a_string, "Hello");
```

- Не се проверява дали дължината на стринговата променлива е достатъчна за да се запише “”Hello”
- Използване на функцията strncpy(променлива, стринг, брой символи)
- 9 символа от a_string_variable ще се копират в another_string

```
char another_string[10];
strncpy(another_string, a_string_variable, 9);
```

# Проверка за еднаквост

- За еднаквост не може да се ползва == (няма да се получи грешка, но резултатът е некоректен)
- Използва се strcmp(низ1, низ2)

```
if (strcmp(c_string1, c_string2))
   cout << "The strings are NOT the same.";
else
   cout << "The strings are the same.";
```

- Функцията връща 0 (лъжа) ако стринговете съвпадат и истина, ако не съвпадат
- cstring.h съдържа горните функции

# Функции за дължина на стринга

- Други функции:

```
strlen("dobedo")
```

```
char string_var[20] = "The rain";
strcat(string_var, "in Spain");
```

- Дължината на стринга в примера е 6.
- В string_var се получава The rainin Spain (внимавайте с празните интервали, предвиждайте къде да се поставят)

# Списък на функции за данни тип C String

| Function | Description | Cautions |
|---|---|---|
| `strcpy(Target_String_Var, Src_String)` | Copies the C-string value *Src_String* into the C-string variable *Target_String_Var*. | Does not check to make sure *Target_String_Var* is large enough to hold the value *Src_String*. |
| `strncpy(Target_String_Var, Src_String, Limit)` | The same as the two-argument `strcpy` except that at most *Limit* characters are copied. | If *Limit* is chosen carefully, this is safer than the two-argument version of `strcpy`. Not implemented in all versions of C++. |
| `strcat(Target_String_Var, Src_String)` | Concatenates the C-string value *Src_String* onto the end of the C string in the C-string variable *Target_String_Var*. | Does not check to see that *Target_String_Var* is large enough to hold the result of the concatenation. |
| `strncat(Target_String_Var, Src_String, Limit)` | The same as the two-argument `strcat` except that at most *Limit* characters are appended. | If *Limit* is chosen carefully, this is safer than the two-argument version of `strcat`. Not implemented in all versions of C++. |
| `strlen(Src_String)` | Returns an integer equal to the length of *Src_String*. (The null character, `'\0'`, is not counted in the length.) | |
| `strcmp(String_1, String_2)` | Returns 0 if *String_1* and *String_2* are the same. Returns a value < 0 if *String_1* is less than *String_2*. Returns a value > 0 if *String_1* is greater than *String_2* (that is, returns a nonzero value if *String_1* and *String_2* are different). The order is lexicographic. | If *String_1* equals *String_2*, this function returns 0, which converts to *false*. Note that this is the reverse of what you might expect it to return when the strings are equal. |
| `strncmp(String_1, String_2, Limit)` | The same as the two-argument `strcat` except that at most *Limit* characters are compared. | If *Limit* is chosen carefully, this is safer than the two-argument version of `strcmp`. Not implemented in all versions of C++. |

# Вход/изход на данни тип C String

- Могат да се използват директно с оператор<<

```
cout << news << " Wow.\n";
```

- news е стрингова променлива, а Wow. – стрингова константа
- Използване на оператор >> но с внимание. Той не приема празни интервали, табулации, стопове. Всяко четене се спира при първия срещнат празен интервал или enter.

```
char a[80], b[80];
cout << "Enter some input:\n";
cin >> a >> b;
cout << a << b << "END OF OUTPUT\n";
```

```
Enter some input:
Do be do to you!
DobeEND OF OUTPUT
```

- Използване на getline

# Пример за използване на функцията getline

```
char a[80];
cout << "Enter some input:\n";
cin.getline(a, 80);
cout << a << "END OF OUTPUT\n";
```

```
Enter some input:
Do be do to you!
Do be do to you!END OF OUTPUT
```

```
char short_string[5];
cout << "Enter some input:\n";
cin.getline(short_string, 5);
cout << short_string << "END OF OUTPUT\n";
```

```
Enter some input:
dobedowap
dobeEND OF OUTPUT
```

Четат се 4 символа,
последният за '\0'

# Конвертиране на стрингове в числа

- "1234" не е число, а стринг

```
int x = atoi("657");
```

sets the value of x to 657, and

```
double y = atof("12.37");
```

sets the value of y to 12.37.

Ако аргументът не е стринг с цифри, се връща стойност 0.
Библиотеката е stdlib.h или cstdlib.h
**atoi("#37")** връща 0 (alphabetic to integer)
**atol(string)** – long; **atof(string)** - double

```
void read_and_clean(int& n)
{
    using namespace std;
    const int ARRAY_SIZE = 6;
    char digit_string[ARRAY_SIZE];

    char next;
    cin.get(next);
    int index = 0;
    while (next != '\n')
    {
        if ( (isdigit(next)) && (index < ARRAY_SIZE - 1) )
        {
            digit_string[index] = next;
            index++;
        }
        cin.get(next);
    }
    digit_string[index] = '\0';

    n = atoi(digit_string);
}
```

# Стандартен string клас

- Този клас позволява типът string да се третира като останалите базови типове
- Необходима библиотека

```
#include <string>
using namespace std;
```

- Класът string позволява стринговите стойности да се разглеждат като стойности от прост тип. Прилага се: =,+ (конкатенация)

```
s3 = s1 + s2;
```

- Ако броят символи на s1 и s2 е по-голям от този на s3, дължината на s3 автоматично се увеличава.
- Начална инициализация

```
string phrase;
string noun("ants");
```

```
string noun("ants");
string noun = "ants";
```

# Пример за използване на клас string

```
//Demonstrates the standard class string.
#include <iostream>
#include <string>
using namespace std;

int main( )
{
    string phrase;
    string adjective("fried"), noun("ants");
    string wish = "Bon appetit!";

    phrase = "I love " + adjective + " " + noun + "!";
    cout << phrase << endl
         << wish << endl;

    return 0;
}
```

*Initialized to the empty string*

*Two ways of initializing a string variable*

```
I love fried ants!
Bon appetit!
```

- Оператори за вход изход << и >> като за другите типове данни.
- При >> се игнорират празните интервали.

```
string s1, s2;
cin >> s1;
cin >> s2;
```

```
May the hair on your toes grow long and curly!
```

s1 –>May; s2->the

```
string line;
cout << "Enter a line of input:\n";
getline(cin, line);
cout << line << "END OF OUTPUT\n";
```

```
Enter some input:
Do be do to you!
Do be do to you!END OF OUTPUT
```

# Различни варианти на функцията getline

```
string line;
cout << "Enter a line of input:\n";
getline(cin, line);
```

```
string line;
cout << "Enter some input:\n";
getline(cin, line, '?');
```

- Чете докато срещне нулев символ; чете докато срещне символа ?

```
string s1, s2;
getline(cin, s1) >> s2;
```

```
cin >> s2;
```

- Първо се изпълнява getline(cin,s1), след това cin>>s2

```
int n;
string line;
cin >> n;
getline(cin, line);
```

```
42
Hello hitchhiker.
```

При този фрагмент ще настъпи грешка, защото getline(cin,line) ще приеме като стойност '\n' след въвеждане на числото 42. Използвайте cin.ignore (1000,'\n') за да се премахнат '\n'.

# Стрингови процеси

- Могат да се извършват същите операции, които са характерни са C string данните
- Стрингът може да се обработва поелементно, като всеки масив от символи

```
//Demonstrates using a string object as if it were an array.
#include <iostream>
#include <string>
using namespace std;

int main( )
{
    string first_name, last_name;

    cout << "Enter your first and last name:\n";
    cin >> first_name >> last_name;

    cout << "Your last name is spelled:\n";
    int i;
    for (i = 0; i < last_name.length( ); i++)
    {
        cout << last_name[i] << " ";
        last_name[i] = '-';
    }
    cout << endl;
    for (i = 0; i < last_name.length( ); i++)
        cout << last_name[i] << " "; //Places a "-" under each letter.
    cout << endl;

    cout << "Good day " << first_name << endl;
    return 0;
}
```

```
Enter your first and last name:
John Crichton
Your last name is spelled:
C r i c h t o n
- - - - - - - -
Good day John
```

За промяна на третия елемент на стринга str

```
str.at(2)='X';
```

или

```
str[2]='X';
```

# Функции за работа със стрингове при клас string

| Example | Remarks |
|---|---|
| Constructors | |
| `string str;` | Default constructor creates empty `string` object `str`. |
| `string str("sample");` | Creates a `string` object with data `"sample"`. |
| `string str(a_string);` | Creates a `string` object `str` that is a copy of `a_string`; `a_string` is an object of the class `string`. |
| Accessors | |
| `str[i]` | Returns read/write reference to character in `str` at index `i`. Does not check for illegal index. |
| `str.at(i)` | Returns read/write reference to character in `str` at index `i`. Same as `str[i]`, but this version checks for illegal index. |
| `str.substr(position, length)` | Returns the substring of the calling object starting at `position` and having `length` characters. |
| `str.length( )` | Returns the length of `str`. |
| Assignment/modifiers | |
| `str1 = str2;` | Initializes `str1` to `str2`'s data. |
| `str1 += str2;` | Character data of `str2` is concatenated to the end of `str1`. |
| `str.empty( )` | Returns *`true`* if `str` is an empty string; *`false`* otherwise. |
| `str1 + str2` | Returns a string that has `str2`'s data concatenated to the end of `str1`'s data. |
| `str.insert(pos, str2);` | Inserts `str2` into `str` beginning at position `pos`. |
| `str.erase(pos, length);` | Removes substring of size `length`, starting at position `pos`. |
| Comparison | |
| `str1 == str2  str1 != str2` | Compare for equality or inequality; returns a Boolean value. |
| `str1 < str2  str1 > str2`<br>`str1 <= str2  str1 >= str2` | Four comparisons. All are lexicographical comparisons. |
| Finds | |
| `str.find(str1)` | Returns index of the first occurrence of `str1` in `str`. If `str1` is not found then the special value `string::npos` is returned. |
| `str.find(str1, pos)` | Returns index of the first occurrence of string `str1` in `str`; the search starts at position `pos`. |
| `str.find_first_of(str1, pos)` | Returns the index of the first instance in `str` of any character in `str1`, starting the search at position `pos`. |
| `str.find_first_not_of (str1, pos)` | Returns the index of the first instance in `str` of any character not in `str1`, starting the search at position `pos`. |

# Примерна програма за проверка дали една дума е палиндром

```
//Test for palindrome property.
#include <iostream>
#include <string>
#include <cctype>
using namespace std;

void swap(char& v1, char& v2);
//Interchanges the values of v1 and v2.

string reverse(const string& s);
//Returns a copy of s but with characters in reverse order.

string remove_punct(const string& s, const string& punct);
//Returns a copy of s with any occurrences of characters
//in the string punct removed.

string make_lower(const string& s);
//Returns a copy of s that has all uppercase
//characters changed to lowercase, other characters unchang

bool is_pal(const string& s);
//Returns true if s is a palindrome, false otherwise.

int main( )
{
    string str;
    cout << "Enter a candidate for palindrome test\n"
         << "followed by pressing Return.\n";
    getline(cin, str);

    if (is_pal(str))
         cout << "\"" << str + "\" is a palindrome.";
    else
         cout << "\"" << str + "\" is not a palindrome.";
    cout << endl;

    return 0;
}

void swap(char& v1, char& v2)
{
    char temp = v1;
    v1 = v2;
    v2 = temp;
}
```

```
string reverse(const string& s)
{
    int start = 0;
    int end = s.length( );
    string temp(s);

    while (start < end)
    {
       end--;
       swap(temp[start], temp[end]);
       start++;
    }

    return temp;
}

//Uses <cctype> and <string>
string make_lower(const string& s)
{
    string temp(s);
    for (int i = 0; i < s.length( ); i++)
        temp[i] = tolower(s[i]);

    return temp;
}
string remove_punct(const string& s, const string& punct)
{
   string no_punct; //initialized to empty string
   int s_length = s.length( );
   int punct_length = punct.length( );

   for (int i = 0; i < s_length; i++)
   {
        string a_char = s.substr(i,1); //A one-character string
        int location = punct.find(a_char, 0);
        //Find location of successive characters
        //of src in punct.

     if (location < 0 || location >= punct_length)
        no_punct = no_punct + a_char; //a_char not in punct, so keep it
   }
   return no_punct;
}
```

# Примерна програма за проверка дали една дума е палиндром

```
//uses functions make_lower, remove_punct
bool is_pal(const string& s)
{
    string punct(",;:.?!'\" "); //includes a blank
    string str(s);
    str = make_lower(str);
    string lower_str = remove_punct(str, punct);

    return (lower_str == reverse(lower_str));
}
```

```
Enter a candidate for palindrome test
followed by pressing Return.
Madam, I'm Adam.
"Madam, I'm Adam." is a palindrome.
```

```
Enter a candidate for palindrome test
followed by pressing Return.
Radar
"Radar" is a palindrome.
```

```
Enter a candidate for palindrome test
followed by pressing Return.
Am I a palindrome?
"Am I a palindrome?" is not a palindrome.
```