King Saud University

College of Computer & Information Sciences

Computer Science Department

CSC114 – Procedural Programming

Spring 2009

Chapter 7.

String and character functions

Lecturer: *Dr. Chaker JEBARI*

int atoi(const char *s);

- **Header File:** stdlib.h
- **Description:**

  Converts a string pointed to by s to int;

- **Return Value:**

  Returns the converted value of the input string. If the string cannot be converted to a number of the corresponding type (int), returns 0.

size_t strlen(const char *s);

- **Header File:** string.h
- **Description:**
  - Calculates the length of a string.
  - strlen calculates the length of s.
- **Return Value:**

  strlen returns the number of characters in s, not counting the null-terminating character.

char *strcat(char *dest, const char *src);

- **Header File:** string.h
- **Description:**
  - Appends one string to another.
  - strcat appends a copy of src to the end of dest.
  - The length of the resulting string is strlen(dest) + strlen(src).
- **Return Value**

  strcat returns a pointer to the concatenated strings:dest

char *strncat(char *dest, const char *src, size_t maxlen);

- **Header File:** string.h
- **Description:**
  - Appends a portion of one string to another.
  - strncat copies at most maxlen characters of src to the end of dest and then appends a null character.
  - The maximum length of the resulting string is strlen(dest) + maxlen.
- **Return Value:**

  strncat returns dest.

int strcmp(const char *s1, const char *s2);

- **Header File:** string.h
- **Description:** Compares one string to another.
- **Return Value:**

If s1 is... return value is...

1. less than s2 < 0
2. the same as s2 == 0
3. greater than s2 > 0

int strncmp(const char *s1, const char *s2, size_t maxlen);

- **Header File:** string.h
- **Description:** Compares a portion of one string to a portion of another.
- **Return Value:**

strncmp returns an int value based on the result of comparing s1 (or part of it) to s2 (or part of it):

1. < 0 if s1 is less than s2
2. == 0 if s1 is the same as s2
3. > 0 if s1 is greater than s2

char *strcpy(char *dest, const char *src);

- **Header File:** string.h
- **Description:**
  - Copies one string into another.
  - Copies string src to dest, stopping after the terminating null character has been moved.
- **Return Value:**

  strcpy returns dest.

char *strncpy(char *dest, const char *src, size_t maxlen);

- **Header File:** string.h
- **Description:**
  - Copies a given number of bytes from one string into another, truncating or padding as necessary.
  - strncpy copies up to maxlen characters from src into dest, truncating or null-padding dest.
  - The target string, dest, might not be null-terminated if the length of src is maxlen or more.
- **Return Value:** strncpy returns dest.

char *strchr(const char *s, int c);

- **Header File:** string.h
- **Description:**
  - strchr finds the first occurrence of the character c in the string s.
  - The null-terminator is considered to be part of the string.
  - **For example:** strchr(strs,0) returns a pointer to the terminating null character of the string strs.
- **Return Value:** strchr returns a pointer to the first occurrence of the character c in s; if c does not occur in s, strchr returns null.

char *strrchr(const char *s, int c);

- **Header File:** string.h
- **Description:**
  - Scans a string for the last occurrence of a given character.
  - strrchr scans a string in the reverse direction, looking for a specific character.

  strrchr finds the last occurrence of the character c in the string s.
  - The null-terminator is considered to be part of the string.
- **Return Value:**
  - strrchr returns a pointer to the last occurrence of the character c.
  - If c does not occur in s, strrchr returns null.

char *strstr(const char *s1, const char *s2);

- **Header File:** string.h
- **Description:**
  - Scans a string for the occurrence of a given substring.
  - strstr scans s1 for the first occurrence of the substring s2.
- **Return Value:**
  - strstr returns a pointer to the element in s1, where s2 begins (points to s2 in s1).
  - If s2 does not occur in s1, strstr returns null.

char *strdup(const char *s);

- **Header File:** string.h
- **Description:**
  - Copies a string into a newly created location.
  - strdup makes a duplicate of string s, obtaining space with a call to malloc. The allocated space is (strlen(s) + 1) bytes long.
  - The user is responsible for freeing the space allocated by strdup when it is no longer needed.
- **Return Value:** strdup returns a pointer to the storage location containing the duplicated string, or returns null if space could not be allocated.

int isalnum(int c);

- **Header File:** ctype.h
- **Description:**
  - Tests for an alphanumeric character.
  - For the default C locale, c is a letter (A to Z or a to z) or a digit (0 to 9).
- **Return Value:**
  - It is a predicate returning nonzero for true and 0 for false.

int isalpha(int c);

- **Header File:** ctype.h
- **Description:** Classifies an alphabetical character.
- **Return Value:**

  isalpha returns nonzero if c is a letter.

int isdigit(int c);

- **Header File:** ctype.h
- **Description:**

  Tests for decimal-digit character.

- **Return Value:**

  isdigit returns nonzero if c is a digit.

int islower(int c);

- **Header File:** ctype.h
- **Description:**

  Tests for lowercase character.

- **Return Value:**

  islower returns nonzero if c is a lowercase letter.

int ispunct(int c);

- **Header File:** ctype.h
- **Description:**

  Tests for punctuation character.

- **Return Value:**

  ispunct returns nonzero if c is a punctuation character.

int isspace(int c);

- **Header File:** ctype.h
- **Description:**

  Tests for space character.

- **Return Value:**

  isspace returns nonzero if c is a space, tab, carriage return, new line, vertical tab, formfeed (0x09 to 0x0D, 0x20), or any other locale-defined space character.

int isupper(int c);

- **Header File:** ctype.h
- **Description:**

  Tests for uppercase character.

- **Return Value:**

  isupper returns nonzero if c is an uppercase letter.

int tolower(int c);

- **Header File:** ctype.h
- **Description:**
  - Translates characters to lowercase.
  - tolower is a function that converts an integer ch (in the range EOF to 255) to its lowercase value (a to z; if it was uppercase, A to Z).
  - All others are left unchanged.
- **Return Value:**

  tolower returns the converted value of ch if it is uppercase; it returns all others unchanged.

int toupper(int c);

- **Header File:** ctype.h

- **Description:**

  - Translates characters to uppercase.

  - toupper is a function that converts an integer ch (in the range EOF to 255) to its uppercase value (A to Z; if it was lowercase, a to z).

  - All others are left unchanged.

- **Return Value:**

  toupper returns the converted value of ch if it is lowercase; it returns all others unchanged.