



CMSC 106 Lecture Set #9 - Strings

Set Started:
Monday, October 25, 2010



C-Strings

- Definition
 - An array of characters
 - Where the used portion is terminated by a null character
- <string.h>
 - Library that acts on C-strings
 - Most will crash if given something that does not fit the definition above
- Creating and Initializing a string
char name1[4] = {'J','a','n','\0'};
char name2[6] = "Plane";
- Characters, strings and numeric values are all different
length of the string and the sizeof operator
 - sizeof operator tells the size of the variable or type
 - strlen uses the definition of C-string to find number of used characters

"0" ≠ '0' ≠ 0



Input and Output

- Output
 - %s in printf format string
 - puts() function takes a string as the only argument
- Input
 - dangerous to use %s in scanf or to use gets() function
 - char *fgets(char *buffer, int bufferSize, FILE *stream);
 - read a line into buffer (at most bufferSize-1 characters)
 - null byte added at end of buffer
 - reads from stream – for standard input just type **stdin** as the name of the stream
 - returns NULL on error or end of file
 - on success returns pointer to the space where you read into (here called the buffer)



Strings

- Zero or more characters followed by null char `'\0'`
 - also called NUL
 - not counted as part of string
 - `string.h` defines prototypes for string routines
- Copying Strings
 - `size_t strlen(char const *str);`
 - returns count of characters in `str`
 - up to but not including the null character
 - `char *strcpy(char *dst, char const *src, size_t len);`
 - copy `src` to `dst`
 - copy until `'\0'` in `src` or at most `len` characters
 - pad extra characters with `'\0'`
 - Safety tip: `dst[len-1] = '\0';` to force termination of new string
 - `char *strncat(char *dst, char const *src, size_t len);`
 - append `src` onto the end of `dst`
 - always appends NUL to end of `dst` string



String Functions

- Comparison
 - `int strcmp(char const *s1, char const *s2, size_t len);`
 - returns 0 if string equal up to `len`
 - returns a negative value if `s1` is less than `s2`
 - returns a positive value if `s1` is greater than `s2`
- Searching
 - `char *strchr(char const *str, int ch);`
 - `char *strrchr(char const *str, int ch);`
 - finds the first occurrence of `ch` in `str`
 - `strrchr` finds the last occurrence
 - returns NULL if not found
 - `char *strstr(char const *s1, char const *s2);`
 - find the first occurrence of `s2` in `s1`



String Functions Examples

```
char string[] = "this is a test string";
char *ans;
int length;

length = strlen(string);
/* returns 21 */
ans = strchr(string, 'h');
/* returns string + 1 */
ans = strrchr(string, 't');
/* returns string + 16 */
ans = strstr(string, "test");
/* returns string + 10 */
```



Character Functions

Prototypes in ctype.h

- Classifying characters
 - parameter is int, but it's a character
 - `int isspace(int ch);`
 - returns true if `ch` is ' ', '\n', '\t', form feed, or carriage return
 - `int isdigit(int ch);`
 - returns true if its 0 through 9
 - `int islower(int ch);` and `isupper(int ch);`
 - return true if it's a-z for islower and A-Z and isupper
 - `int isalpha(int ch);`
 - returns true if it's a-z or A-Z
 - `int isalnum(int ch);`
 - returns true if it's a-z or A-Z or 0-9
- Transformation
 - `int toupper(int ch);` `int tolower(int ch);`
 - converts to upper/lower case



References for C Language Libraries

<http://www.utas.edu.au/infosys/info/documentation/C/CStdLib.html>

<http://en.wikipedia.org/wiki/String.h>



typedef

- Allows you to define a new type
- Format:
`typedef whatItIs whatYouWantToCallIt;`
- For example:
`typedef int Bool;`
- Array example:
`typedef char MyString[MAX];`
