ECS30: string & `char` comparisons, operations

Dr. Rob Gysel
1/23/17

Overview

- strings in C
  - `char` arrays
  - Printing with `%s`
- String operations: indexing, terminating, `strlen`
- Comparing strings: `strcmp`
- More on `<ctype.h>`

Strings in C

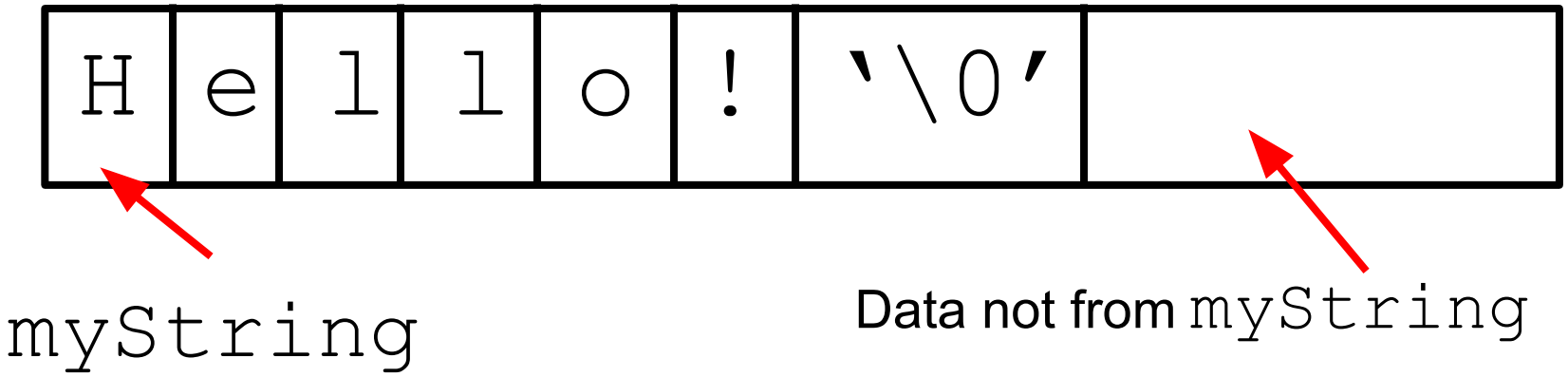
- No “string” type
- Represented by `char` *arrays*
- Need to know when they end: *terminated* with *null terminator character* `'\0'`

Strings

- Declaring strings:

```
char myString[10] = "Hello!"
```

- In memory:

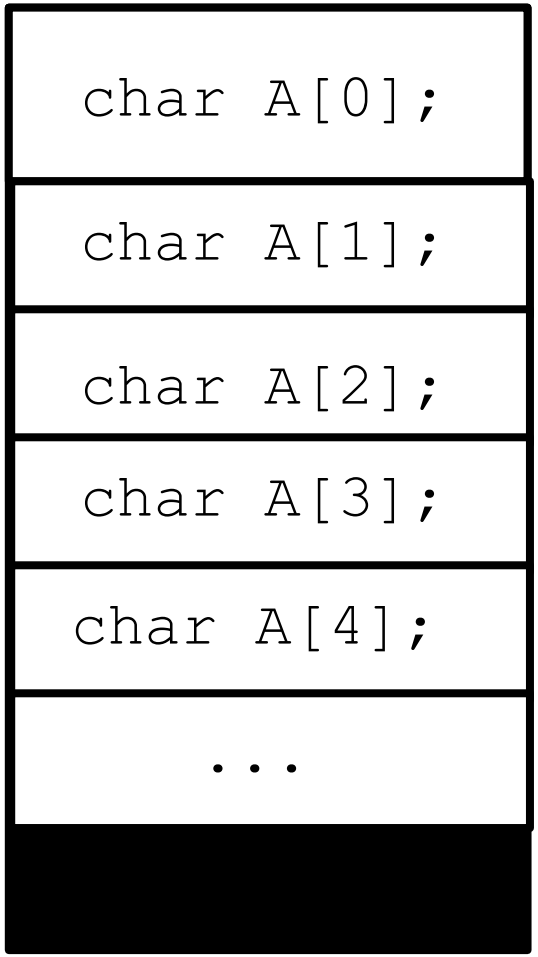


Indexing strings with `A[i]`

- *Arrays* are a collection of the same data type
- Contiguous in memory
- Syntax:

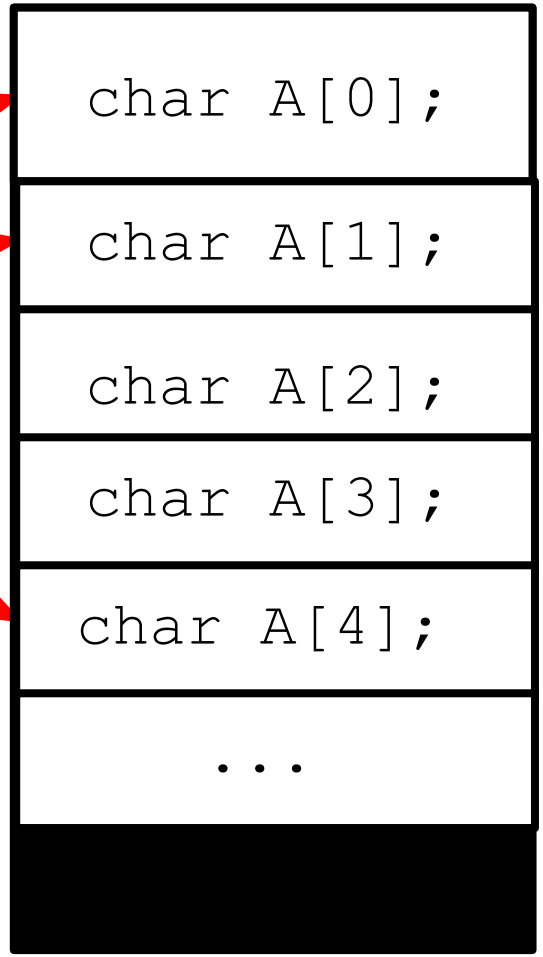
  `char A[5];`

  - 5-`char` array
  - `A[0]` first element
  - `A[4]` last element



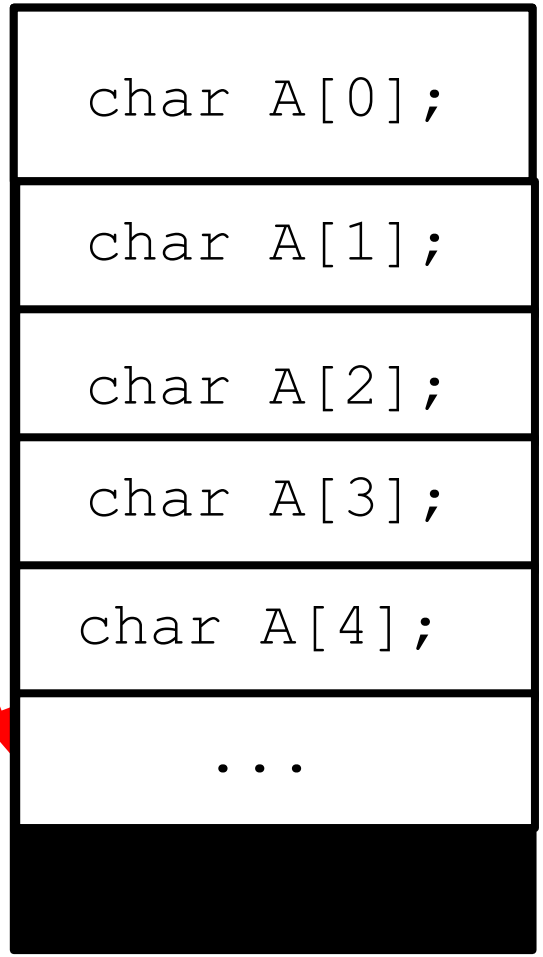
Indexing strings with `A[i]`

- Indices start at 0
  - `A[0]` is the first element
  - `A[1]` is the second element
  - Last element of `A`
  - Other data not associated with `A`



Indexing strings with `A[i]`

- Accessing past an array's bounds results in a *Segmentation Fault*
- Does not always crash
  - Can silently continue without exiting program
  - This leads to **very hard-to-find bugs**



Strings, `printf`, `scanf`

- Use `%s` for the format string in scanf and printf to read / write a string
  - scanf reads until whitespace is found, adds `'\0'` at end
  - printf prints until `'\0'` is found

Standard library `<string.h>` functions

1. Concatenate strings: `strcat`
2. Copy string: `strcpy`
3. Compare strings: `strcmp`
4. Length: `strlen` (# chars before `'\0'`)
5. Tokenize (parse) strings: `strtok`

Copying strings

With `strcpy(dest, src)`:

```
int main() {

  char string1[3] = “hi”,
string2[4] = “hey”;

  strcpy(string2, string1);

}
```

Copying strings

Q1: Why 3?

With `strcpy(dest, src)`:

```
int main() {

  char string1[3] = “hi”,
string2[4] = “hey”;

  strcpy(string2, string1);

}
```

Copying strings

Q2: Swap ok?

With `strcpy(dest, src)`:

```
int main() {

  char string1[3] = “hi”,
string2[4] = “hey”;

  strcpy(string2, string1);

}
```

Comparing Strings

The wrong way to compare:

```
char *string1, *string2;

  if (string1 == string2)
```

- compares memory locations (addresses), not strings!

Comparing Strings

Compare with `strcmp` instead:

```
if (strcmp(string1, string2) == 0){
… }
```

- returns 0 if equal, ex.

`strcmp("hi", "hi")` is `0`

`strcmp("hi", "hey")` is not `0`

Comparing Strings

If strings differ `strcmp` compares *alphabetically:*

`string1 < string2` if `string1` comes before `string2` alphabetically

- “hey” < “hi”
- “ban” < “banana”

Comparing Strings

`strcmp` returns positive / negative int based on alphabetical order:

- `0 < strcmp("hey", "hi")` because "hey" < "hi"
- `0 > strcmp("hi", "hey")` because "hi" > "hey"

Standard library `<ctype.h>` functions

1. `isalpha` T/F is alphabetic `char`
2. `isalnum` T/F is alphanumeric `char`
3. `isdigit` T/F is digit `char`
4. `islower / isupper`
   T/F is lowercase / uppercase `char`
5. `ispunct` T/F is punctuation `char`

Standard library `<ctype.h>` functions

1. `isalpha` T/F is alphabetic `char`
2. `isalnum` T/F is alphanumeric `char`
3. `isdigit` T/F is digit `char`
4. `islower / isupper`
   T/F is lowercase / uppercase `char`
5. `ispunct` T/F is punctuation `char`
6. `isspace` T/F is whitespace `char`

Whitespace characters

Whitespace characters: spaces, tabs, new lines, etc.

- `char space = ' ';`
- `char tab = '\t';`
- `char newline = '\n';`
- `char carriagereturn = '\r';`

Common escape sequences

- `char backslash = '\\';`
- `char doubleQuote = '\"';`
- `char singleQuote = '\'';`
- `char nullTerminator = '\0';`

^ Always need to end string