C – Variables, Strings and Loops

Karthik Dantu

Ethan Blanton

Computer Science and Engineering

University at Buffalo

kdantu@buffalo.edu

Administrivia

- How many of you did the assigned reading?
- How many of you have your VM working?
- How many of you compiled and rand the hello world program?
- How many of you looked into the shell commands posted on Piazza?

- ASIDE: Recompile hello world and introduce man pages

Types

- C is a typed language

  Each variable has a type, and is declare

  Every value assigned to that variable must match the type

- Compiler will automatically convert between some types

Valid

```
int x = 0;
float y = 0;
x = 37;
y = x;
```

Invalid

```
int x = 0;
x = “hello, world”
```

The image part with relationship ID rId15 was not found in the file.

The image part with relationship ID rId15 was not found in the file.

C Types

- Lots of Types; for now, consider:
  - `int`: Integers of a convenient size for the computer (32-bit)
  - `char`: Characters (typically 8-bit integers)
  - `double`: Double-precision floating point numbers
- Array types

Declared with square brackets: []

- `char a[]`: Array of characters
- `int scores[200]` : An array of exactly 200 integers

The image part with relationship ID rId15 was not found in the file.

The image part with relationship ID rId15 was not found in the file.

Declaring Variables

- Variables are declared by stating their name and type
- Variables retain their type *while they are in scope*
- Various modifiers can be applied to variables
- In particular `const` declares a variable to be constant

```
int x; /* x is an integer
*/

double num; /* num is a
floating-point double */


const int pi=3.14; /* pi
is an integer constant*/
```

The image part with relationship ID rId15 was not found in the file.

The image part with relationship ID rId15 was not found in the file.

Scopes

- Variables in C have scope
- A variable cannot be used out of scope
- Variables declared outside any block ({})

  Are usually global – can be accessed by any code

  Are file-local with the modifier `static` – they can be accessed by any code in this file
- Variables declared inside any block ({})

  Come into scope when declared

  Are valid until the end of scope (})

The image part with relationship ID rId15 was not found in the file.

The image part with relationship ID rId15 was not found in the file.

Arrays

- C arrays are a series of contiguous memory locations
- Arrays are declared with []. The size is between []
- Arrays can have three "sizes", depending on what's in the []

  Unknown size: Nothing is specified

  Constant size: A constant expression is specified

  Variable size: A run-time computed expression is specified

Arrays – Known Size

- Array sizes specify how many elements are in the array

```
int x[32];
int matrix [32][16];
```

- C **does not remember** the array's size
- Therefore, illegal accesses are not caught

```
int x[4];
x[10234] = 0; /* Whoops. */
```

Unknown Array Sizes

- Unknown array sizes are limited in use
- They often appear as arguments to functions (as in `main`())
- An array of unknown size cannot be declared normally
- Sizes are required for multidimensional arrays

```
void func(int matrix [][3][2]);
```

Array Indexing

- C array indices start from 0
- An array of size 10 contains elements 0 through 9
- Arrays can be dereferenced with []

```
int array[10];
int i=7;

array[4] = 0;
array[i] = 0;
array[i+1]=0;
```

The image part with relationship ID rId15 was not found in the file.

The image part with relationship ID rId15 was not found in the file.

Static Initializers

- An array can be initialized all at once at declaration

```
int array [10] = { 0, 3, 5, 0, 0, 1, 0, 0, 2, 0
};
```

- This is called a static initializer
- Static initializers can only be used at declaration

```
int array [3];
array = { 1, 3, 5 }; /* syntax error */
```

C Strings

- C strings are an array of `char`
- A C string consists of:

  the characters in the string, followed by

  a zero byte (the ASCII NUL character) (NUL terminator).
- The zero byte is idiomatically written as ‘`0`’
- Strings, like arrays do not have an associated length
- You can count the number of `char` to know how long the string is

The image part with relationship ID rId15 was not found in the file.

The image part with relationship ID rId15 was not found in the file.

String Examples

- Strings are represented as a series of characters between double quotes
- Strings can be declared as follows

```
char str [] = "Hello";
/* str = { 'H', 'e', 'l', 'l', 'o', '\0' } */
```

- Like arrays, such an assignment is possible only at declaration
- After declaration, strings must be copied into arrays

```
char str [32];
strncpy(str , 32, "Hello"); /* See man 3 strncpy */
```

String Functions In C

- There are many string functions in the C library.
- Most of them are defined in `<string.h>`
- Some useful examples:

  `strlen()`: Compute the length of a string by counting bytes

  `strncpy()`: Copy a string until its NUL character

  `strncat()`: Concatenate one string to another

  `strstr()`: Search for one string inside another

The image part with relationship ID rId15 was not found in the file.

The image part with relationship ID rId15 was not found in the file.

Character Constants

- C code is in ASCII encoding
- ASCII contains Latin characters, numbers and punctuation
- An ASCII character can be converted into an integer

```
char c = 'A'; /* 65 */
int i = 'B'; /* 66 */
```

- Each character of a string can be assigned in this manner

```
char str [] = "emacs";
/* Give it the respect it deserves */
str [0] = 'E';
```

The `for` loop

- The C for loop is the common loop construct
- It allows looping over almost anything

```
for ( initialization ; condition; increment) {
   body;
}
```

- It translates to a more traditional while loop (with caveats)

```
initialization;
while (condition) {
   body;
   increment;
}
```

Looping Over Arrays

- A common use of for loop is to loop over arrays

```
int array[ARRAYSZ ];
for (int i = 0; i < ARRAYSZ; i++) {
/* Use array[i] */
}
```

- Array size needs to be known or calculated

Modifying Control Flow

- Two keywords control loop execution:
  `break`
  `continue`
- The continue statement will immediately:
  Execute the increment statement
  Start the body over at the top
- The break statement will immediately end the loop.

Looping Over Strings

- Just like arrays, we can use for to loop over strings
- We can look for the NUL terminator instead of needing to know array size

```
for (int i = 0; str[i] != '\0'; i++) {
   /* use str[i] */
}
```

- No need to compute string length in this example

The image part with relationship ID rId15 was not found in the file.

The image part with relationship ID rId15 was not found in the file.

Loop Example

We will develop strlen() together

Summary

- C is a typed language - every variable has a type
- Variable values must match the type
- Variables have scope, and cannot be used outside that scope
- Arrays are contiguous memory locations
- Array syntax uses []
- C strings are arrays of characters
- Every C string is terminated with a zero byte
- For loop syntax
- For loops are very flexible

Required Readings

Last Class

- K&R: 1.6, 1.7, 1.9

Next Class

- K&R: 1.9, 1.10, 2.1, 2.2, 2.3, 2.4