Review of C programming

Glenn Bruns
CSUMB

Lecture Objectives

When this lecture is finally over, you should be able to:

- Write and compile C programs
- Explain some of the differences between C and C++

C

- Created at Dennis Ritchie at Bell Labs between 1969 and 1973
- History:
  - BCPL, Univ. of Cambridge, 1967
  - B, Ken Thompson and Dennis Ritchie, Bell Labs, 1969
  - C, Dennis Ritchie, Bell Labs, 1973
  - (C++, Bjarne Stroustrup, Bell Labs, 1983)
- C standards:
  - C89 (aka C90)
  - C99
  - C11 (2011)

Hello world

Problem: output "hello, world!" to the screen

```
#include <stdio.h>

// hello world in C

int main(void) {
   printf("hello, world!\n");

   // main program should return a value
   return 0;
}
```

Run it

```
$ gcc -o ex1 ex1.c
$ ./ex1
hello, world!
$
```

Note: by default, gcc uses the gnu90 "standard", which is the C89 standard with GNU-specific extensions

Input/Output

Problem: input number from user and display it

```
#include <stdio.h>

#define MAX_BUF 4

// getting input from stdin

int main(void) {
   char s[MAX_BUF];

   printf("enter a number: ");
   char *status = fgets(s, MAX_BUF, stdin);
   printf("you entered ---%s---\n", s);

   return 0;
}
```

There is no cin or cout in C

Run it

```
$ gcc -o ex2 ex2.c
$
$ ./ex2
enter a number: 123
you entered ---123---
$ ./ex2
enter a number: 12
you entered ---12
---
$ ./ex2
enter a number: 1234
you entered ---123---
```

Can you explain this?

Loops

Problem: loop over values 0 to 9 and display them

```
#include <stdio.h>

// loop from 0 to 9

int main(void) {
   int i;
   for (i = 0; i < 10; i++) {
      printf("i = %d\n", i);
   }

   return 0;
}
```

Run it

```
$ make ex3
cc     ex3.c   -o ex3
$ ./ex3
i = 0
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
```

How did I compile using make?

Using 'break' and 'continue' in loops

Problem: do nothing on value 2, exit loop on value 8

```
#include <stdio.h>

int main(void) {
   int i;
   for (i = 0; i < 10; i++) {
      if (i == 2) {
         continue;
      }
      if (i == 8) {
         break;
      }
      printf("i = %d\n", i);
   }
   return 0;
}
```

Run it

```
$ ./ex3a
i = 0
i = 1
i = 3
i = 4
i = 5
i = 6
i = 7
```

Arrays

```
#include <stdio.h>
#define BLEN 6

// find the max value in a non-empty array

int main(void) {
   int i;
   float b[BLEN] = {0.2, 1.5, -7.3, 4.6, 5.1, 2.1};
   float max = b[0];
   for (i = 1; i < BLEN; i++) {
      if (b[i] > max) {
         max = b[i];
      }
   }
   printf("max = %0.2f\n", max);

   return 0;
}
```

Run it

```
$ ./ex4
max = 5.10
$
```

Functions

Problem: write a function to find the max value of a float array

```
#include <stdio.c>

// function to find the max value in an array

// return the max value of the first n elements of float array x
// n must be greater than 0, and the length of x must be at least n
float array_max(float x[], int n) {
   int i;
   float max = x[0];
   for (i = 1; i < n; i++) {
      if (x[i] > max) {
         max = x[i];
      }
   }
   return max;
}

int main() {
   float x[] = {0.2, 1.5, -7.3, 4.6, 5.1, 2.1};
   printf("max = %0.2f\n", array_max(x,6));
   return 0;
}
```

Pointers

```
// experiments with pointers

int main() {
   int b = 0;
   int *bp;
   int c[] = {3,2,1};

   printf("b = %d\n", b);
   bp = &b;                        // get address of b
   printf("b = %d\n", *bp);        // get value at address bp

   printf("address of c[0] = %p\n", c);
   printf("address of c[0] = %p\n", &(c[0]));

   // pointer arithmetic == array indexing
   printf("c[2] = %d\n", c[2]);
   printf("c[2] = %d\n", (int)*(c + 2));

   return 0;
}
```

Try it

```
$ ./ex6
b = 0
b = 0
address of c[0] = 0xbfd8b60c
address of c[0] = 0xbfd8b60c
c[2] = 1
c[2] = 1
$
```

Strings

```
#include <stdio.h>
#include <string.h>

// experiments with strings
int main(void) {
   char *s = "doozy";
   printf("%s has length %d\n", s, strlen(s));
   printf("s[0]: %c, s[5]: %d\n", s[0], s[5]);

   char s1[] = "woozy";
   s1[3] = 't';
   printf("new string: %s\n", s1);

   if (s == "wooty") {                        // what about this?
      printf("s is wooty");
   }
   if (strcmp(s, "doozy") == 0) {
      printf("s is doozy\n");
   }

   printf("compare abc and xyz: %d\n", strcmp("abc", "xyz"));
   printf("compare xyz and abc: %d\n", strcmp("xyz", "abc"));
   return 0;
}
```

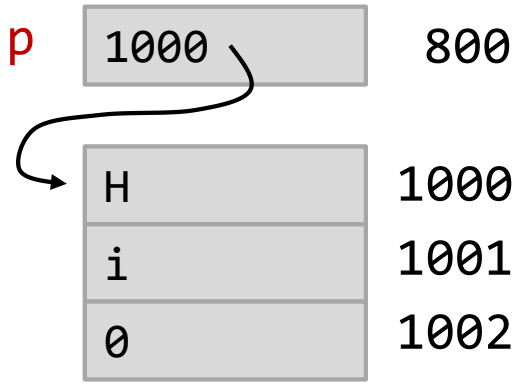
Try it

```
$ ./ex7
string: doozy has length 5
s[0]: d, s[5]: 0
new string: wooty
s is doozy
compare abc and xyz: -1
compare xyz and abc: 1
```

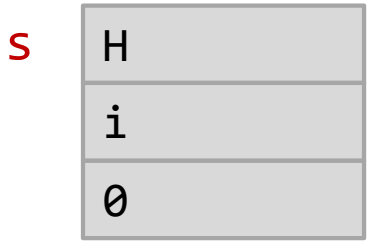
Arrays vs. pointers

```
#include<stdio.h>

int main(void) {
   char *p   = "Hi";  // p is a pointer; points to a "literal string"
   char  s[] = "Hi";  // s is a character array, initialized with "Hi"

   printf("%s %c\n", p, p[1]);    // output is 'Hi i'
   printf("%s %c\n", s, s[1]);    // output is 'Hi i'

   printf("%d\n", sizeof(p));  // how many bytes for p?
   printf("%d\n", sizeof(s));  // how many bytes for s?
}
```

picture of memory for p:



picture of memory for s:



Structs

```
#include <stdio.h>
#include <stdlib.h>

// experiments with structs

typedef struct node {        // node in a linked list
   int val;
   struct node *next;
} NODE;

int main() {
   NODE *head, *n1;

   n1 = (NODE *)malloc(sizeof(NODE));
   n1->val = 10;
   n1->next = NULL;

   head = (NODE *)malloc(sizeof(NODE));
   head->val = 5;
   head->next = n1;

   printf("first val: %d, second val: %d\n", head->val, head->next->val);
   return 0;
}
```

Run it

```
$ ./ex8
first val: 5, second val: 10
$
```

Booleans

```
#include <stdio.h>
#include <stdbool.h>

// C has no boolean type

int main() {
   printf("Comparison of 1 and 2: %d\n", 1 == 2);

   if (0) {
      printf("0 means 'true'\n");
   }
   if (5) {
      printf("5 means 'true'\n");
   }

   // stdbool.h defines true and false
   if (true) {
      printf("Thanks, stdbool!\n");
   }
   return 0;
}
```

Run it

```
$ ./ex9
Comparison of 1 and 2: 0
5 means 'true'
Thanks, stdbool!
$
```

C interprets 0 as false; all other ints as true

Summary

- C was invented around 1973
- Some differences between C and C++:
  - C is not object-oriented (has no classes)
  - C++ uses cin, cout for I/O; C uses printf, scanf
  - C has no built-in boolean type or class
  - C++ uses new/delete for memory allocation; C uses malloc/free