C programming language

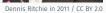
Jeremy Iverson

College of Saint Benedict & Saint John's University

origins

• Dennis Ritchie and Brian Kernighan creators of C circa 1972







Brian Kernighan in 2012 / CC BY 2.0

comparison

Java	С
object-oriented	procedural
interpreted	compiled
String	char array
condition (boolean)	condition (int)
garbage-collected	no memory management
references	pointers
exceptions	error codes

- · in Java, everything is a method that is called on an object
- · in C, everything is a function
- in Java, source code is compiled to byte code, which is then interpreted by Java VM
- in C, source code is compiled into binary machine code
- in Java, String is a class
- in C, a string is just an array of char values which ends with the char '\0'
- in Java, the Java VM takes care of deallocating memory used
- in C, any memory you allocate, you must also deallocate

hello, world

```
#include <stdio.h>

int main() {
   printf("hello, world\n");
   return 0;
}
```

```
$ gcc -o helloworld helloworld.c
$ ./helloworld
hello, world
```

• The tradition of using the phrase "Hello, world!" as a test message was influenced by an example program in the seminal book *The C Programming Language*

conditions

· under what conditions will each of the following be execute?

```
if (x) {
    /* ??? */
}

if (x-y) {
    /* ??? */
}

if (x=y) {
    /* ??? */
}

/* ??? */
}
```

- x != 0
- x != y
- y != 0

add evens

 create program called add_even.c that adds all the even numbers between 1 and 100 and prints the sum

```
#include <stdio.h>

int main(int argc, char * argv[]) {
   printf("(%d) %s:%s\n", argc, argv[0],
        argv[1]);
   return 0;
}
```

 modify add_even.c to get maximum value from the command-line instead of hard-coded as 100

printf / scanf

- printf() interprets variables and prints character representations to standard out (usually the terminal)
- scanf() scans characters from standard in (usually the terminal) and interprets them for storage in variables

```
#include <stdio.h>

int main() {
   int i;
   scanf("%d", &i);
   return 0;
}
```

 scanf requires you to pass the address of the variable, so that its value can be changed

echo

- modify **helloworld.c** to ask user for an input and then print
- · change its name to echo.c

it back

```
$ ./echo
Enter a string to echo: hello, world
hello.
```

why did it print hello, instead of hello, world?

pointers

• a pointer is a variable whose value is a memory address

```
int i = 0x1A;
int * ip = &i;
```

- $\upbeta \mathbf{i}$ evaluates to the address where the variable \mathbf{i} is stored in memory
- i is an int, so ip is a pointer to an int

pointers cont.

```
printf("0x%X\n", i);  /* 0x1A */
printf("0x%#X\n", &i); /* 0x12A0 */
printf("0x%#X\n", ip); /* 0x12A0 */
printf("0x%#X\n", &ip); /* 0x???????? */
```

• so how can we use the pointer, ip, to access the value of i?

pointer dereference

- *ptr will
 - 1. treat the value of **ptr** as a memory address
 - 2. get the bytes of data located at that memory address
 - 3. interpret those bytes according to the type of pointer that ptr is

11

```
printf("0x%X\n", *ip); /* 0x1A */
```

the C compiler is "smart enough" to "know" that + X really means add
 X * sizeof(*ip) to ip

pointer dereference

- *ptr will
 - 1. treat the value of ptr as a memory address
 - 2. get the bytes of data located at that memory address
 - 3. interpret those bytes according to the type of pointer that ptr is

11

```
printf("0x%X\n", *ip); /* 0x1A */
```

 $\cdot ip[X] = *(ip + X)$

printf("0x%X\n", ip[0]); /* 0x1A */

the C compiler is "smart enough" to "know" that + X really means add
 X * sizeof(*ip) to ip

pointers cont.

pointers cont.

```
char * cp = "hello, world";
 · cp is a pointer to a char
0x00004C80 | h | e |
0x??????? | 00 | 00 | 4C | 80
   printf("%c\n", cp[0]); /* h */
   printf("%c\n", cp[4]); /* o */
   printf("%c\n", *(cp+4)); /* 0 */
```

printf("%s\n", cp+7); /* world */
printf("0x%#X\n", cp); /* 0x4C80 */

why not say cp is a pointer to a char array?

practice

```
void swap(int n1, int n2) {
 printf("BEFORE v1=%d, v2=%d\n", v1, v2);
 swap(v1, v2);
  return 0;
```

- what's wrong with this program?
- fix the program so that it correctly swaps the two variables' values

heap memory

 designate a block of memory to store value(s) of a particular data type

· allocates enough consecutive memory for 100 **int** values

heap memory

 designate a block of memory to store value(s) of a particular data type

• release a block of memory back to system to be used elsewhere

```
free(ip);
```

• allocates enough consecutive memory for 100 int values

heap memory cont.

heap memory cont.

file i/o

```
int main(int argc, char * argv[]) {
   return -1;
 if (!fclose(fp))
  return 0;
```

final thoughts

matrices



except where otherwise noted, this worked is licensed under creative commons attribution-sharealike 4.0 international license