CS200 - Computer Organization

C Programming Intro

Aravind Mohan

Allegheny College

September 6, 2022

Last class

- **Reminder:** Workshop files submission due by end of this week.
- **Quiz 1:** In Two Weeks, Tuesday, Sep 20th during class timings.
- **Make sure** Docker and Java Installation issues were resolved.
- **Last week:** Bits and Bytes and activity on (Decimal to Binary) Match Table.

History of C

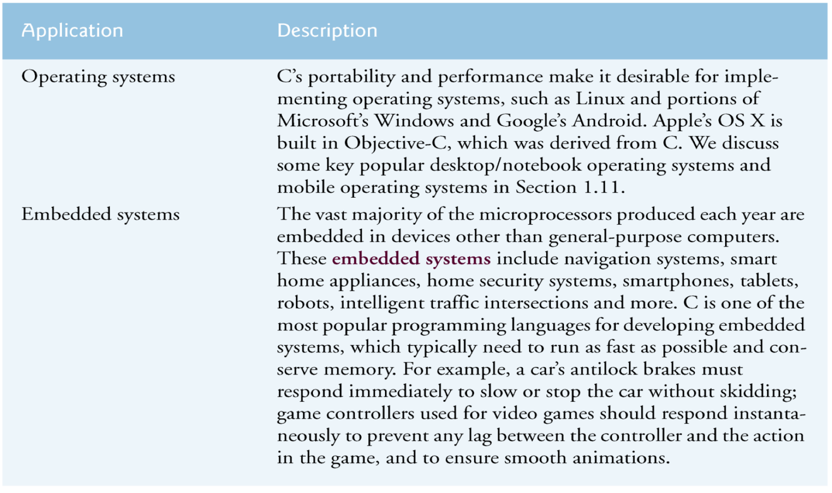
- C **evolved** from two previous languages, BCPL and B.
- Ken Thompson and Dennis M. Ritchie were the most prestigious 1983 **Turing Award** winners. Ken and Dennis modeled C at Bell Labs.
- C initially became widely known as the development language of the **UNIX** operating system.

What is so special about C?

Built for Performance

- C is widely used to develop systems that **demand performance**, such as operating systems, embedded systems, real-time systems and communications systems.

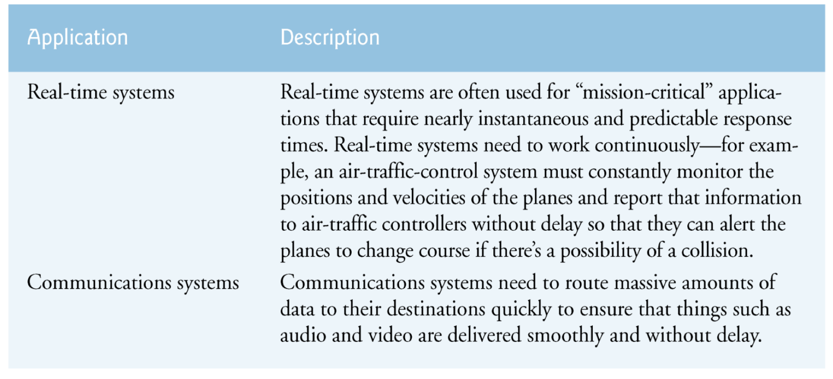
Applications written in C



Application	Description
Operating systems	C's portability and performance make it desirable for implementing operating systems, such as Linux and portions of Microsoft's Windows and Google's Android. Apple's OS X is built in Objective-C, which was derived from C. We discuss some key popular desktop/notebook operating systems and mobile operating systems in Section 1.11.
Embedded systems	The vast majority of the microprocessors produced each year are embedded in devices other than general-purpose computers. These **embedded systems** include navigation systems, smart home appliances, home security systems, smartphones, tablets, robots, intelligent traffic intersections and more. C is one of the most popular programming languages for developing embedded systems, which typically need to run as fast as possible and conserve memory. For example, a car's antilock brakes must respond immediately to slow or stop the car without skidding; game controllers used for video games should respond instantaneously to prevent any lag between the controller and the action in the game, and to ensure smooth animations.

Figure 1: Some popular performance-oriented C applications

Applications written in C (contd)



Application	Description
Real-time systems	Real-time systems are often used for “mission-critical” applications that require nearly instantaneous and predictable response times. Real-time systems need to work continuously—for example, an air-traffic-control system must constantly monitor the positions and velocities of the planes and report that information to air-traffic controllers without delay so that they can alert the planes to change course if there’s a possibility of a collision.
Communications systems	Communications systems need to route massive amounts of data to their destinations quickly to ensure that things such as audio and video are delivered smoothly and without delay.

Figure 2: Some popular performance-oriented C applications

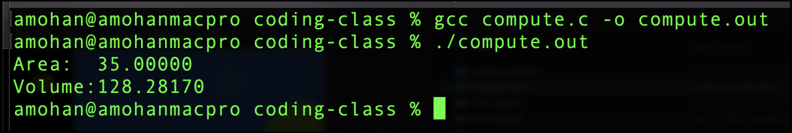
Math computation in C

```
// Computes volume and area of
// triangle, volume of cone.
#include <stdio.h>
#define PI 3.1415926535
int main(void) {
  double height, base, area, volume, radius;
  height = 10; base = 7;
  radius = base/2.0;
  // area of a triangle
  area = 0.5*base*height;
  // volume of cone
  volume = PI*radius*radius*height/3.0;
  printf("Area:%10.5f\nVolume:%9.5f\n",area,volume);

  return 0;
}
```

Refer compute.c in coding folder

Math Computation in C (contd)



```
amohan@amohanmacpro coding-class % gcc compute.c -o compute.out
amohan@amohanmacpro coding-class % ./compute.out
Area:  35.00000
Volume:128.28170
amohan@amohanmacpro coding-class % 
```

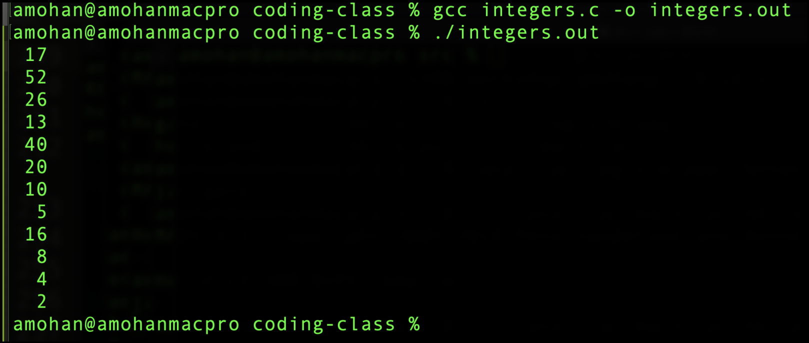
- %10.5f means 10 spaces wide and 5 places after the decimal point.
- Refer K&R page 13 for more formats.

Logical programming in C

```
// Altering numbers using a formula.
#include <stdio.h>
int main(){
  int n = 17;
  while (n > 1){
    printf("%3d\n",n);
    if (n % 2 == 0){
      n = n/2;
    }
    else{
      n = 3*n+1;
    }
  }
  return 0;
}
```

Refer integers.c in coding folder

Logical programming in C (contd)



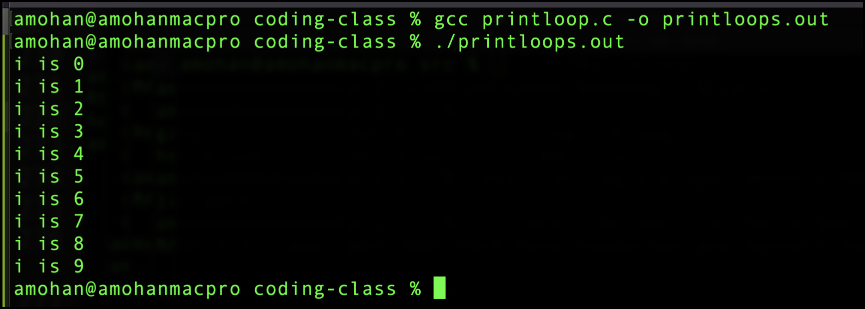
```
amohan@amohanmacpro coding-class % gcc integers.c -o integers.out
amohan@amohanmacpro coding-class % ./integers.out
 17
 52
 26
 13
 40
 20
 10
  5
 16
  8
  4
  2
amohan@amohanmacpro coding-class %
```

Loops in C

```
// Prints sequence of numbers using a for loop.
#include <stdio.h>
int main(){
  for (int i = 0; i < 10; i++){
    printf("i is %d\n",i);
  }
  return 0;
}
```

Refer printloop.c in coding folder

Loops in C (contd)



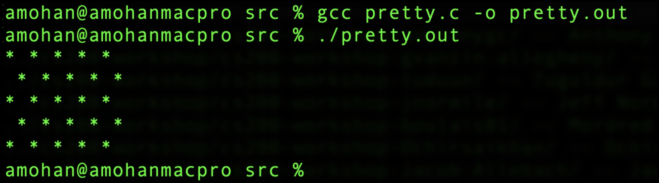
```
amohan@amohanmacpro coding-class % gcc printloop.c -o printloops.out
amohan@amohanmacpro coding-class % ./printloops.out
i is 0
i is 1
i is 2
i is 3
i is 4
i is 5
i is 6
i is 7
i is 8
i is 9
amohan@amohanmacpro coding-class % 
```

Nested for loop in C

```
// Draws a pretty picture.
#include <stdio.h>
int main(){
  char star = ’*’, space = ’ ’;
  for (int row = 0; row < 5; row++){
    for (int col = 0; col < 5; col++){
      if (row%2 == 0){
        printf("%c%c",star,space);
      }
      else{
        printf("%c%c",space,star);
      }
    }
    printf("\n");
  }
  return 0;
}
```

Refer pretty.c in coding folder

Nested for loop in C (contd)

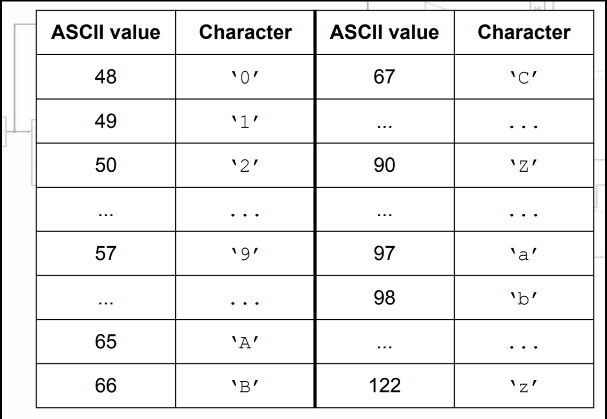


```
amohan@amohanmacpro src % gcc pretty.c -o pretty.out
amohan@amohanmacpro src % ./pretty.out
* * * * *
 * * * * *
* * * * *
 * * * * *
* * * * *
amohan@amohanmacpro src %
```

ASCII Representation

- How do computers do things at the machine level?
- Primitive ”character-at-a-time” processing.
- What happens when your code does ”i = 3821;” ?

ASCII Vs Integers



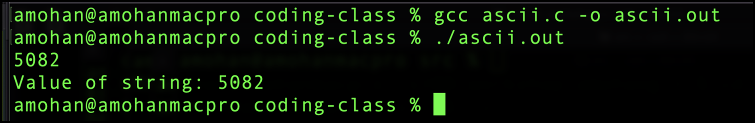
ASCII value	Character	ASCII value	Character
48	‘0’	67	‘C’
49	‘1’	...	. . .
50	‘2’	90	‘Z’
...	. . .	...	. . .
57	‘9’	97	‘a’
...	. . .	98	‘b’
65	‘A’	...	. . .
66	‘B’	122	‘z’

Convert ASCII To Integers

```
#include <stdio.h>
int main(){
  int c; /* input character--assumed to be a digit */
  int digit; /* the decimal digit corresponding to c */
  /* value of the input string as an int */
  int value = 0;
  while ((c = getchar()) != ’\n’) {
    /* Check for error in input */
    if (c < ’0’ || c > ’9’){
      printf("Error--non-digit in input\n");
      break;
    }
    /* Convert ASCII to digit */
    digit = c - ’0’;
    /* Combine with previous digits */
    value = 10 * value + digit;
  }
  printf("Value of string: %d\n",value);
  return 0;
}
```

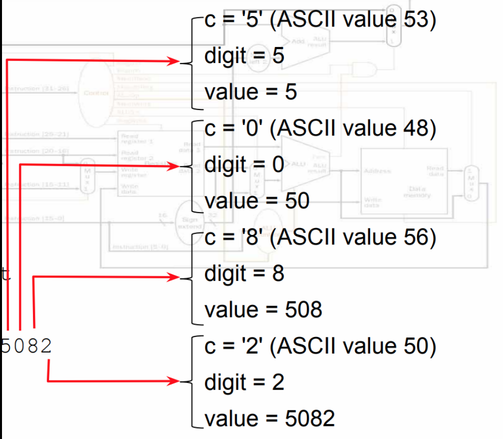
Refer ascii.c in coding folder

Convert ASCII To Integers in C (contd)



```
amohan@amohanmacpro coding-class % gcc ascii.c -o ascii.out
amohan@amohanmacpro coding-class % ./ascii.out
5082
Value of string: 5082
amohan@amohanmacpro coding-class % 
```

Convert ASCII To Integers(contd)



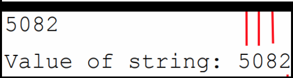
5082

c = '5' (ASCII value 53)
digit = 5
value = 5

c = '0' (ASCII value 48)
digit = 0
value = 50

c = '8' (ASCII value 56)
digit = 8
value = 508

c = '2' (ASCII value 50)
digit = 2
value = 5082



```
5082
Value of string: 5082
```

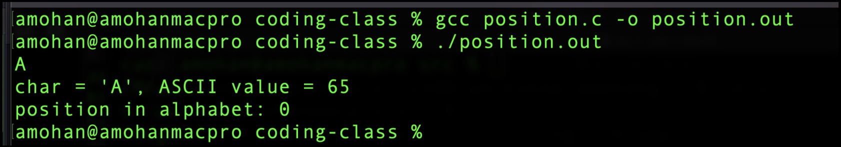
Convert ASCII To Integers(contd)

In C, characters are treating like integers. All of the following are perfectly legal (and even make sense):

```
#include <stdio.h>
int main(){
  int c, position;
  /* read in a character, e.g, A */
  c = getchar();
  printf("char = ’%c’, ASCII value = %d\n",c,c);
  position = c - ’A’;
  printf("position in alphabet: %d\n",position);
  return 0;
}
```

Refer position.c in coding folder

Convert ASCII To Integers in C (contd)



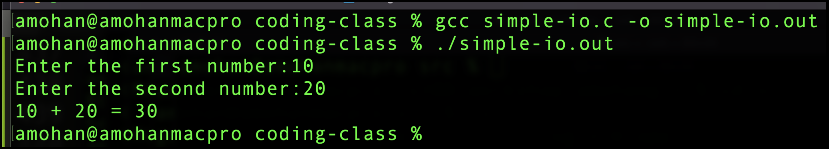
```
amohan@amohanmacpro coding-class % gcc position.c -o position.out
amohan@amohanmacpro coding-class % ./position.out
A
char = 'A', ASCII value = 65
position in alphabet: 0
amohan@amohanmacpro coding-class %
```

A Simple I/O program in C

```
#include <stdio.h>
int main()
{
    int a, b, c;
    printf("Enter the first number:");
    scanf("%d", &a);
    printf("Enter the second number:");
    scanf("%d", &b);
    c = a + b;
    printf("%d + %d = %d\n", a, b, c);
    return 0;
}
```

Refer simple-io.c in coding folder

A Simple I/O program in C (contd)



```
amohan@amohanmacpro coding-class % gcc simple-io.c -o simple-io.out
amohan@amohanmacpro coding-class % ./simple-io.out
Enter the first number:10
Enter the second number:20
10 + 20 = 30
amohan@amohanmacpro coding-class %
```

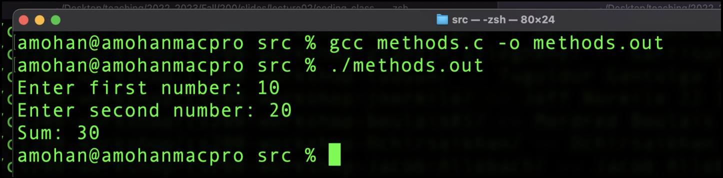
Methods in C

```
#include <stdio.h>
int add(int x, int y){
  int z = x + y;
  return z;

}
int main()
{
    int first, second = 0;
    printf("Enter first number: ");
    scanf("%d", &first);
    printf("Enter second number: ");
    scanf("%d", &second);
    int result = add(first, second);
    printf("Sum: %d\n", result);
    return 0;
}
```

Refer methods.c in coding folder

Methods in C (contd)



```
amohan@amohanmacpro src % gcc methods.c -o methods.out
amohan@amohanmacpro src % ./methods.out
Enter first number: 10
Enter second number: 20
Sum: 30
amohan@amohanmacpro src % 
```

- **How is Pass by value different from Pass by reference?**

Reading Assignment

1. PH chapter 01 - section 1.3
2. KR chapter 01 - section 1.1 - 1.5

Questions?

Please ask your Questions!