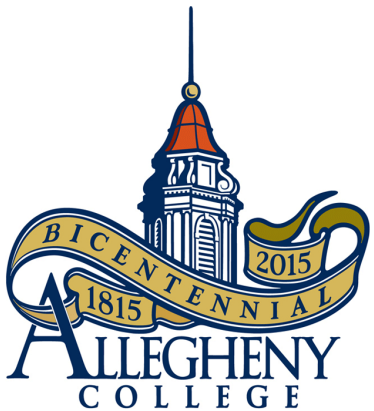
CS200 - Computer Organization

C Programming Intro

Aravind Mohan

Allegheny College

August 31, 2021



Last class

- **Reminder:** Lab 01 due by end of this week.
- Docker 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

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

Preprocessor commands

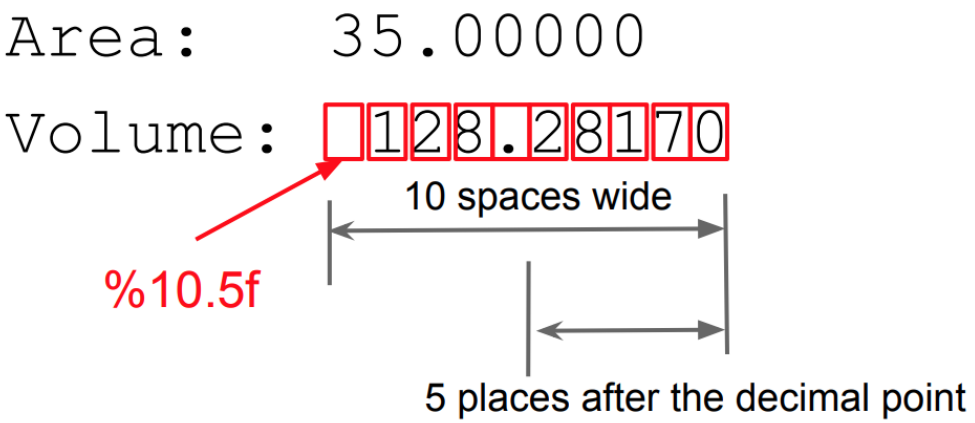
Same as "int main()" (int is the default return type)

Just like Java!

Format specifiers:

refer compute.c in coding folder

Math Computation in C (contd)



(See K&R page 13 for more on formats)

Logical programming in C

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

Just like Java!

refer integers.c in coding folder

Output Displayed:

```
17
52
26
13
40
20
10
 5
16
 8
 4
 2
```

Loops in C

```
#include <stdio.h>
int main() {
    for (int i = 0; i < 10; i++) {
        printf("i is %d\n",i);
    }
}
```

```
notjava.c: In function 'main':
notjava.c:3:5: error: 'for' loop initial declarations are
only allowed in C99 mode
     for (int i = 0; i < 10; i++) {
```

refer printloop.c in coding folder

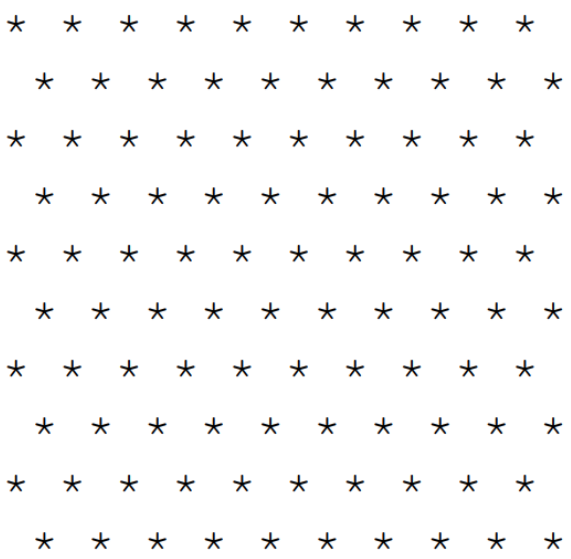
Nested for loop in C

```
// Draws a pretty picture
#include <stdio.h>
int main() {
    char star = '*', space = ' ';
    int row, col;
    for (row = 0; row < 10; row++) {
        for (col = 0; col < 10; col++) {
            if (row%2 == 0) {
                printf("%c%c",star,space);
            }
            else {
                printf("%c%c",space,star);
            }
        }
        printf("\n");
    }
}
```

refer pretty.c in coding folder

Nested for loop in C(contd)

Output Displayed:



ASCII Representation

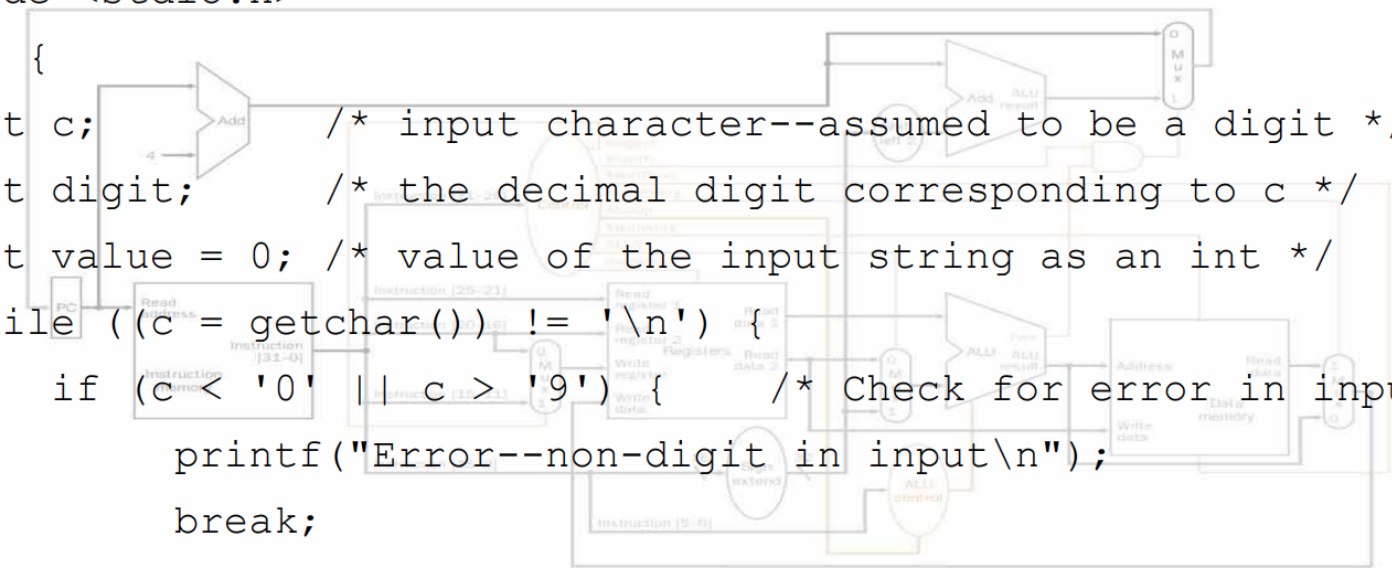
- How do computers do things at the machine level?
- Primitive ”character-at-a-time” processing.
- What happens when your code does ”i = 3821;” ?

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'

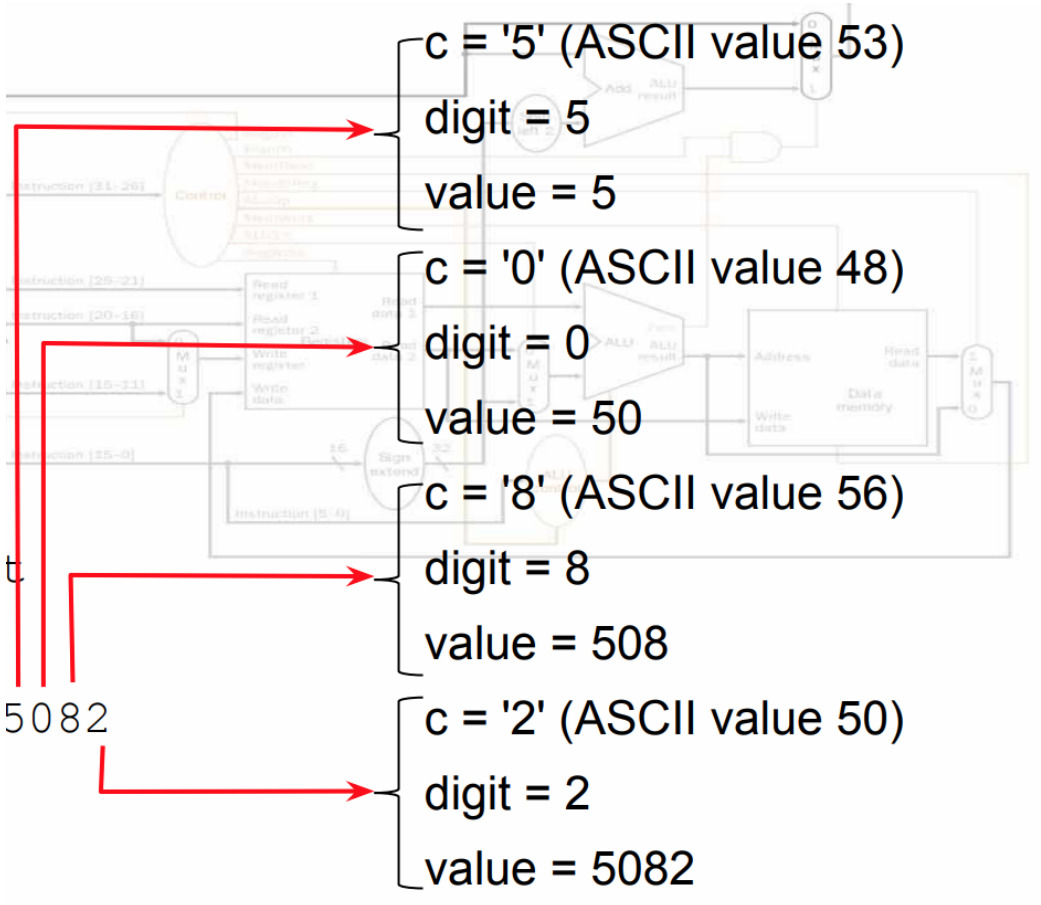
Convert ASCII To Integers

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



refer ascii.c in coding folder

Convert ASCII To Integers(contd)




```
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):

```
int c, position;
c = getchar(); /* read in a character, e.g, 'A' */
printf("char = '%c', ASCII value = %d\n",c,c);
position = c - 'A';
printf("position in alphabet: %d\n",position);
```

Output Displayed:

```
char = 'A', ASCII value = 65
position in alphabet: 0
```

refer position.c in coding folder

A simple IO 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 simpleio.c in coding folder

Reading Assignment

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

Class activity

1. Form groups of three and discuss!
2. **Provide a programming solution to implement ToUpper and ToLower methods?**
3. Post the solution in the ”class-activity” channel on the Slack site.
4. The solution doesn’t necessarily need to be a code file, it could be pseudocode!
5. A variation of this exercise may be included in the next lab.

Questions

Do you have any questions from this class discussion?