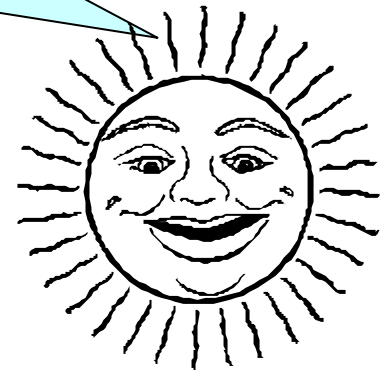


CSE1301  
Computer Programming  
Lecture 4:  
*C Primitives I*

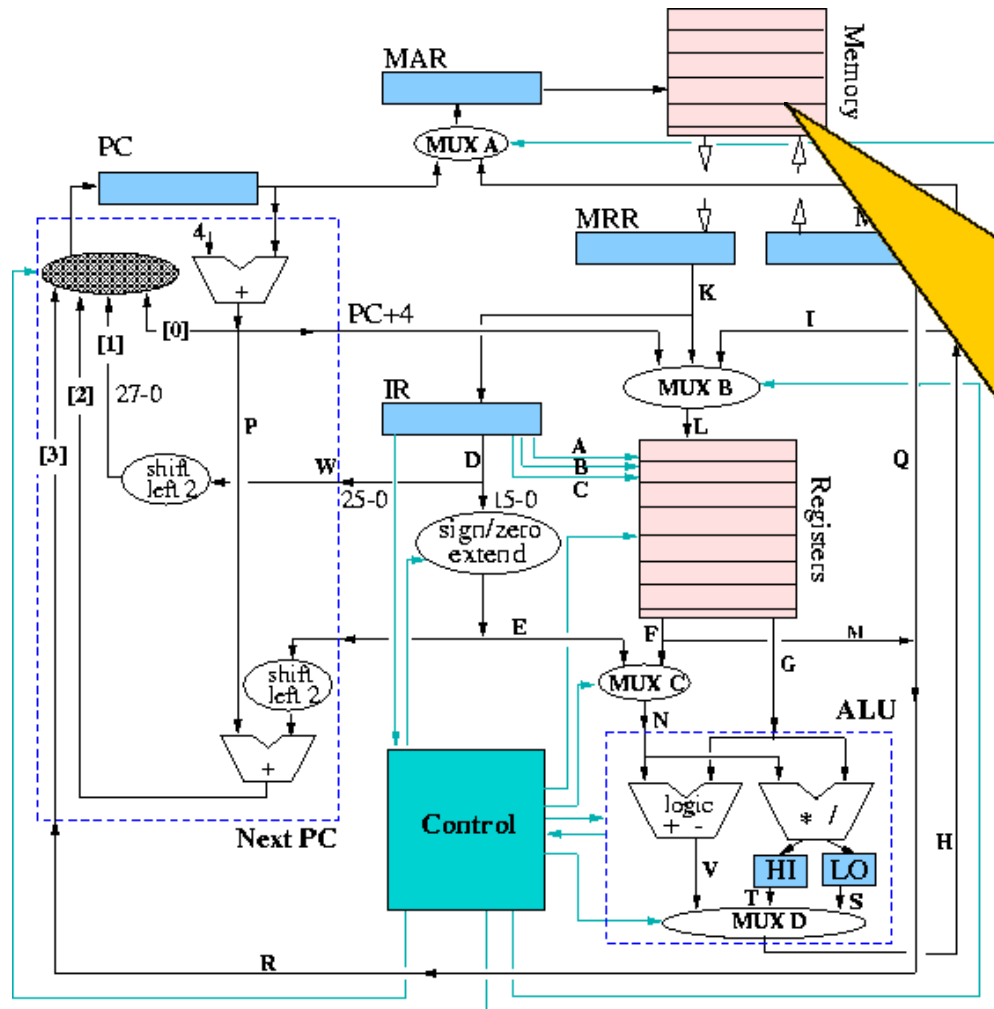
# Topics

- History of C
- Structure of a C program
- Values and variables
- Expressions
- Function calls
- Comments

```
printf("Hello World");
```



# Machine Language



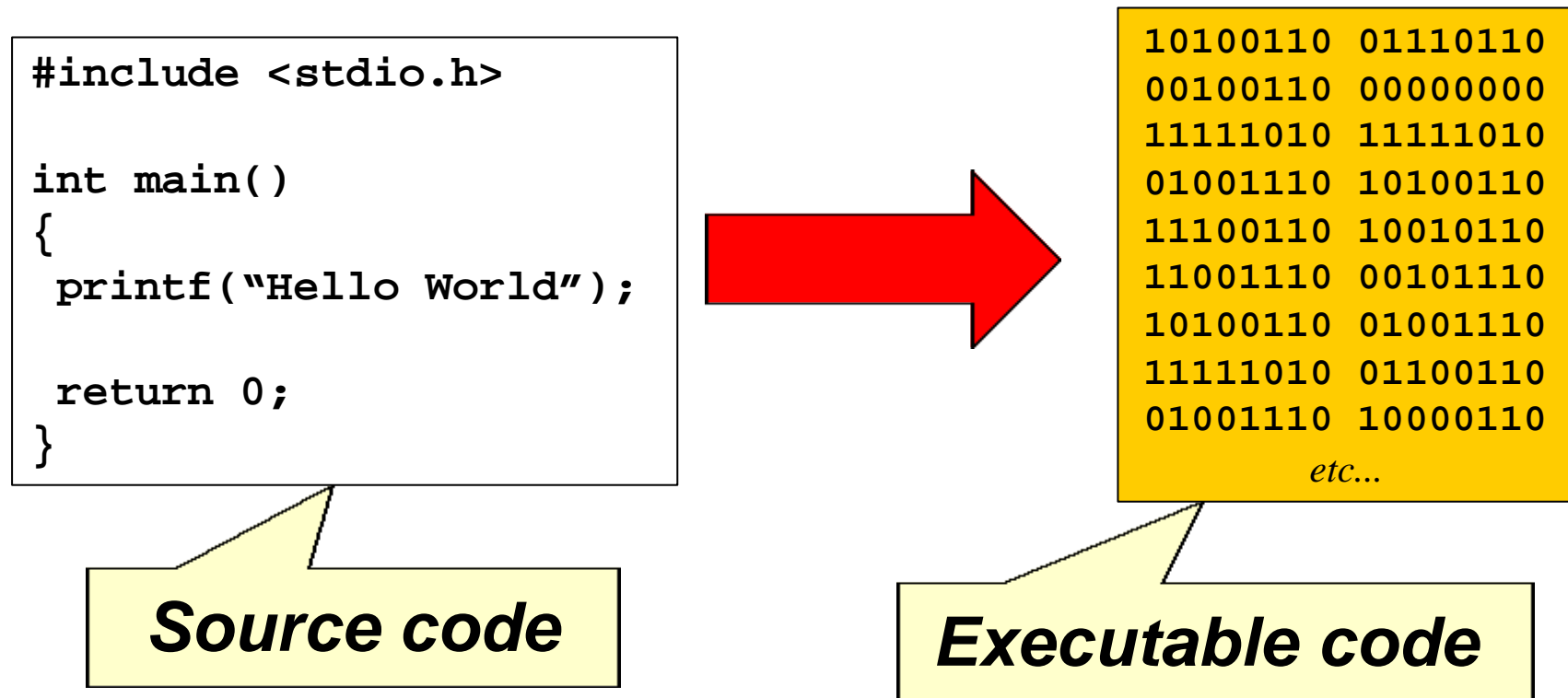
```

10100110 01110110
00100110 00000000
11111010 11111010
01001110 10100110
11100110 10010110
11001110 00101110
10100110 01001110
11111010 01100110
01001110 10000110
etc...
    
```

# From Algorithms to Programs

- Both are sets of instructions on how to do a task
- Algorithm:
  - talking to humans, easy to understand
  - in plain (English) language
- Program:
  - talking to computer (compiler)
  - can be regarded as a “formal expression” of an algorithm

# High-Level Language



- **Compilers** and **linkers** translate a high level program into executable machine code.

# Why C?

- Flexible language:
  - Structured language
  - Low level activities possible
- Standard library exists, allowing portability
  - See D&D 2/e Appendix B (web links in D&D 3/e)
  - Forouzan & Gilberg Appendix F
- It can produce lean and efficient code
- Wide availability on a variety of computers
- Widely used

# History of C

- **CPL** Combined Programming Language (Barron et al., 1963)
- **BCPL** Basic CPL (Richards, 1969)
- **B** (Thompson, 1970)
- **C** K&R C (Ritchie, 1972)
- **ANSI C** American National Standards Institute C (X3J11, 1989)
- **C99** (JTC1/SC22/WG14, ISO/IEC 9899, 1999)

# Basic Structure of a C Program

Example: Hello World

Algorithm:

output "Hello World!"

C Program:

```
#include <stdio.h>

int main()
{
    printf("Hello World!");

    return 0;
}
```

# Basic Structure of a C Program (cont)

Example: Hello world

C Program:

Includes **standard input/output library** of procedures.

*Read: "Hash-include"*

```
#include <stdio.h>

int main()
{
    printf("Hello World!");

    return 0;
}
```

# Basic Structure of a C Program

Example: Hello World

Curly braces mark the **beginning** and **end** of a block of instructions.

C Program:

```
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```

# Basic Structure of a C Program

Example: Hello World

Instruction (**function call**)  
to output "Hello World"

C Program:

```
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```

# Basic Structure of a C Program

Example: Hello World

C Program

```
#include <stdio.h>

int main()
{
    printf("Hello World");

    return 0;
}
```

“Statements” (lines of instructions) always end with a **semi-colon (;)**

# Example -- Count to 10

Print out numbers 0 to 9

```
set count to 0
while ( count is less than 10 )
{
    output count
    add 1 to count
}
```

```
int main()
{

    return 0;
}
```

# Example -- Count to 10 (cont)

Print out numbers 0 to 9

```
set count to 0
while ( count is less than 10 )
{
    output count
    add 1 to count
}
```

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

```
int main()
{

    return 0;
}
```

# Example -- Count to 10 (cont)

Print out numbers 0 to 9

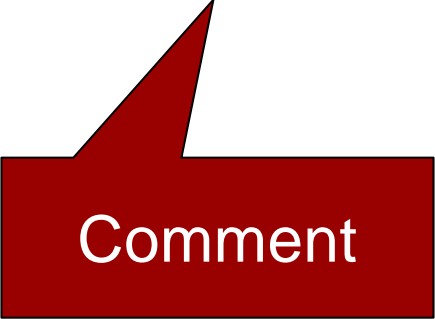
```
set count to 0
while ( count is less than 10 )
{
    output count
    add 1 to count
}
```

```
#include <stdio.h>

/* Print out numbers 0 to 9 */

int main()
{

    return 0;
}
```



Comment

# Example -- Count to 10 (cont)

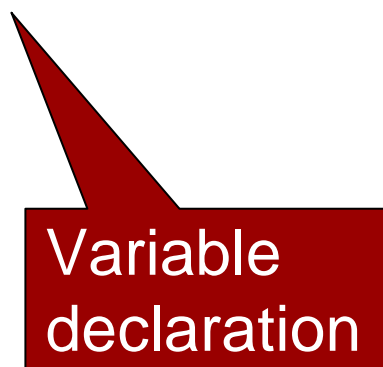
Print out numbers 0 to 9

```
set count to 0
while ( count is less than 10 )
{
    output count
    add 1 to count
}
```

```
#include <stdio.h>

/* Print out numbers 0 to 9 */
int main()
{
    int count;

    return 0;
}
```



Variable  
declaration

# Example -- Count to 10 (cont)

Print out numbers 0 to 9

```
set count to 0
while ( count is less than 10 )
{
    output count
    add 1 to count
}
```

```
#include <stdio.h>

/* Print out numbers 0 to 9 */
int main()
{
    int count;

    count = 0;
    while ( count < 10 )
    {
        printf("%d\n", count);
        count=count+1;
    }
    return 0;
}
```

# Example -- Count to 10 (cont)

Print out numbers 0 to 9

set count to 0

```
while ( count is less than 10 )  
{  
    output count  
    add 1 to count  
}
```

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

```
/* Print out numbers 0 to 9 */
```

```
int main()
```

```
{
```

```
    int count;
```

```
    count = 0;
```

```
    while ( count < 10 )
```

```
    {
```

```
        printf("%d\n", count);
```

Assignment of a value  
(right expression) to a  
variable (left).

```
    }
```

```
}
```

# Example -- Count to 10 (cont)

Print out numbers 0 to 9

```
set count to 0
while ( count is less than 10 )
{
    output count
    add 1 to count
}
```

```
#include <stdio.h>

/* Print out numbers 0 to 9 */
int main()
{
    int count;

    count = 0;
    while ( count < 10 )
    {
        printf("%d\n", count);
        count=count+1;
    }
    return 0;
}
```

No semi-  
colon here!

# Example -- Count to 10 (cont)

Print out numbers 0 to 9

```
set count to 0
while ( count is less than 10 )
{
    output count
    add 1 to count
}
```

```
#include <stdio.h>

/* Print out numbers 0 to 9 */
int main()
{
    int count;

    count = 0;
    while ( count < 10 )
    {
        printf("%d\n", count);
        count=count+1;
    }
    return 0;
}
```

# Example -- Count to 10 (cont)

Print out numbers 0 to 9

```
set count to 0
while ( count is less than 10 )
{
    output count
    add 1 to count
}
```

```
#include <stdio.h>

/* Print out numbers 0 to 9 */
int main()
{
    int count;

    count = 0;
    while ( count < 10 )
    {
        printf("%d\n", count);
        count=count+1;
    }
    return 0;
}
```



Format string

# Example -- Count to 10 (cont)

Print out numbers 0 to 9

```
set count to 0
while ( count is less than 10 )
{
    output count
    add 1 to count
}
```

```
#include <stdio.h>

/* Print out numbers 0 to 9 */
int main()
{
    int count;

    count = 0;
    while ( count < 10 )
    {
        printf("%d\n", count);
        count=count+1;
    }
    return 0;
}
```

## Example -- What's your sign?

Find the sign of a number

output "Enter a number"  
input num

if (num is less than 0)  
then

```
{  
    output num " is -'ve"  
}
```

else

```
{  
    output num " is +'ve"  
}
```

```
#include <stdio.h>  
/* Find the sign of a number */  
int main()  
{  
    float num;  
    printf("Enter a number: ");  
    scanf("%f", &num);  
  
    if ( num < 0 )  
    {  
        printf("%f is -'ve\n", num);  
    }  
    else  
    {  
        printf("%f is +'ve\n", num);  
    }  
    return 0;  
}
```

## Example -- What's your sign? (cont)

Find the sign of a number

output "Enter a number"  
input num

if (num is less than 0)  
then

{  
    output num " is -'ve"  
}

else

{  
    output num " is +'ve"  
}

```
#include <stdio.h>
/* Find the sign of a number */
int main()
{
    float num;

    printf("Enter a number: ");
    scanf("%f", &num);

    if ( num < 0 )
    {
        printf("%f is -'ve\n", num);
    }
    else
    {
        printf("%f is +'ve\n", num);
    }

    return 0;
}
```

## Example -- What's your sign? (cont)

Find the sign of a number

output "Enter a number"

input num

if (num is less than 0)

then

{

    output num " is -'ve"

}

else

{

    output num " is +'ve"

}

```
#include <stdio.h>
/* Find the sign of a number */
int main()
{
    float num;

    printf("Enter a number: ");
    scanf("%f", &num);

    if ( number < 0 )
    {
        printf("%f is -'ve\n", num);
    }
    else
    {
        printf("%f is +'ve\n", num);
    }
    return 0;
}
```

## Example -- What's your sign? (cont)

Find the sign of a number

output "Enter a number"

input num

if (num is less than 0)

then

{

    output num " is -'ve"

}

else

{

    output num " is +'ve"

}

```
#include <stdio.h>
/* Find the sign of a number */
int main()
{
    float num;

    printf("Enter a number: ");
    scanf("%f", &num);

    if ( num < 0 )
    {
        printf("%f is -'ve\n", num);
    }
    else
    {
        printf("%f is +'ve\n", num);
    }
    return 0;
}
```

## Example -- What's your sign? (cont)

Find the sign of a number

output "Enter a number"

input num

if (num is less than 0)

then

{

    output num " is -'ve"

}

else

{

    output num " is +'ve"

}

```
#include <stdio.h>
/* Find the sign of a number */
int main()
{
    float num;

    printf("Enter a number: ");
    scanf("%f", &num);

    if ( num < 0 )
    {
        printf("%f is -'ve\n", num);
    }
    else
    {
        printf("%f is +'ve\n", num);
    }
    return 0;
}
```

# Topics

- ✓ History of C
- ✓ Structure of a C program
- Values and variables
- Expressions
- Function calls
- Comments

# Values and Variables

- Basic Types:
  - Integers
  - Floating point numbers
  - Characters
  - Character Strings

# Basic Types: **int** and **float**

- Integers (**int**)

0    1    1000    -1    -10    666

- Floating point numbers (**float**)

1.0    .1    1.0e-1    1e1

# Basic Types: **char**

- Characters (**char**)

'a' 'z' 'A' 'Z' '?' '@' '0' '9'

- Special Characters: preceded by \

'\n' '\t' '\0' '\'' '\\' *etc.*

# Basic Types: character string

- Character Strings (a string of **char**-s)
- Examples:
  - `"Hi there!"`
  - `"Line 1\nLine 2\nLine 3"`
  - `""`
  - `"\\"`

# Topics

- ✓ History of C
- ✓ Structure of a C program
- ✓ Values and variables

# Reading

- King
  - Chapter 1, 1.1 – 1.2
  - Chapter 2, 2.1
- D&D:
  - Chapter 2, Sections 2.1 to 2.5
- Kernighan & Ritchie
  - Chapter 1, 1.1