start

{
  return 0;
  printf("%d\n", i); //arguments
  for (i=0; i<argc; i++)
    printf("%d\n", argc);
    int i;
}

int main(int argc, char *argv[])

Arguments to main()
Arguments to main()
# include <stdio.h>

Fahrenheit to Celsius

{ printf("Celsius = %\n\n" , celsius);
  celsius = (fahrenheit-32)*5.0/9;
fahrenheit = atof(argv[1]);
  double fahrenheit, celsius;
}

int main(int argc, char *argv[])
{
  # include <stdio.h>

}
Arguments to main()
```c
#include <stdio.h>

double Fahrenheit, Celsius;

{
    printf("Enter Fahrenheit\n");
    scanf("%f", &Fahrenheit);

    Fahrenheit = (Fahrenheit - 32) * 5.0 / 9;
    printf("%f\n", Fahrenheit);
}

int main()
{
    return 0;
}
```

Fahrenheit to Celsius
Argumments to main()
{
    return
    printf("%%d%%d\n", mo, day, year);
    /* output formatted date */
}

void print_date(int mo, int day, int year)
Example
actual parameters are pointers

Call by reference

Formal parameter receives the value of the actual parameter (arrays are an exception)

Array would only be read, not rewritten

Function can not change the value of the actual parameter

Call by value
within the program file. a global variable is defined outside the main function and can be accessed by any function or block that defines it. a local variable is defined within a function or a block and can be accessed only within the function.
The function or variable is visible or accessible.

It is valid to reference the function or variable.

Scope refers to the portion of the program in which

Scope of a function or variable

```c
{ return
  { 
    n--
factores = factores*n;
  }
}

int factores = 1;
// do not call if
+(does not exist)
}

int fact(n)
{
  return 0;
  printf("result is %d\n", s);
  s = fact(t) + fact(t+1);
  int t = 5, s;
}

int main(void)
/* prototype */
#include <stdio.h>

```
Different functions
Same variable name can be used in
Scope of a function or variable
Global vs Local Variable

```c
#include <stdio.h>

int main() {
    int a = 3;
    int b = 4;
    int c = 5;
    int d = 6;

    int z = 2;
    z = z + a;
    printf("%d\n", z);

    return 0;
}

int function() {
    int a = 3;
    if (a == 2) {
        int z = 2;
        printf("%d\n", z);
    }
    return 0;
}
```
Storage Class - 4 types

- **automatic** - key word **auto** - default for local variables
  - Memory set aside for local variables is not reserved when the block in which the local variable was defined is exited.

- **external** - key word **extern** - used for global variables
  - Memory is reserved for a global variable throughout the execution life of the program.

- **static** - key word **static**
  - Requests that memory for a local variable be reserved throughout the execution life of the program. The static storage class does not affect the scope of the variable.

- **register** - key word **register**
  - Requests that a variable should be placed in a high speed memory register.
How can you compute the area of an irregular shape?

Random Numbers

Engineering problems require use of random numbers.

Min, Max, Avg, possible outcomes are equally likely or not.

Rolling a die (1, 2, 3, 4, 5, 6)

Tossing a coin (heads, tails)

What is a random number?
For example, 1 and 100?

What if we need random numbers in another range?

between 1 and 6 by rolling a die.

So we can generate uniform random numbers

same probability of 1/6.

For example, fair die, where each outcome has the

All outcomes are equally likely.

Uniform Random Numbers
Program:

What will be the output if we re-run the same

```c
printf("%d \n", rand()); rand(); rand();
```

What will be the output of the following

2147483647) defined in stdlib.

RAND_MAX is a system dependent constant (e.g.,

generate random numbers between 0 and RAND_MAX

In Standard C Library we have a function rand() to

Uniform Random numbers
In Standard C library, we have a function `rand()`. To give a new seed number `seed` to get a new sequence of pseudo-random numbers, we must use the same sequence of pseudo-random numbers. So, if you give the same `seed` you will always get the same sequence of pseudo-random numbers using the same algorithm. Computers generate pseudo-random numbers using a seed.
```c

return 0;
}

printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n}\n```

```
\[
\text{\begin{tabular}{l}
\text{\texttt{int rand\_int(int a, int b)}}
\end{tabular}}
\]

{ }

\text{return rand()} \cdot (b - a + 1) + a;

\]

\text{\begin{tabular}{l}
\text{x = 10 + rand()} \% 8;
\end{tabular}}

\text{\begin{tabular}{l}
\text{Generate a RN between 10 to 17}
\end{tabular}}

\text{\begin{tabular}{l}
\text{x = rand()} \% 8;
\end{tabular}}

\text{\begin{tabular}{l}
\text{Generate a RN between 0 and 7}
\end{tabular}}

[ a \text{ b} ]

\text{\begin{tabular}{l}
\text{RNS in a specified range [a b]}
\end{tabular}}
{ return (double)(rand() / (RAND_MAX * (b-a) + a)); }

double Rand_Float(double a, double b)

The value is then shifted into range [a, b] by adding a

between 0.0 and b-a

will give a RN

number between 0.0 and 1.0

will give a random

Floating-Point RNs in [a, b]
Macros

```
#define PI 3.14
#define area_tri(base, height) (0.5 * base * height)
```

Examples

- Macro-text replaces macro-name in the program
- `#define macro-name(parameters) macro-text`