```c
#include <stdio.h>

int main () {
   while (1) {
      printf ("%s", "I love to teach");
   };
}
```

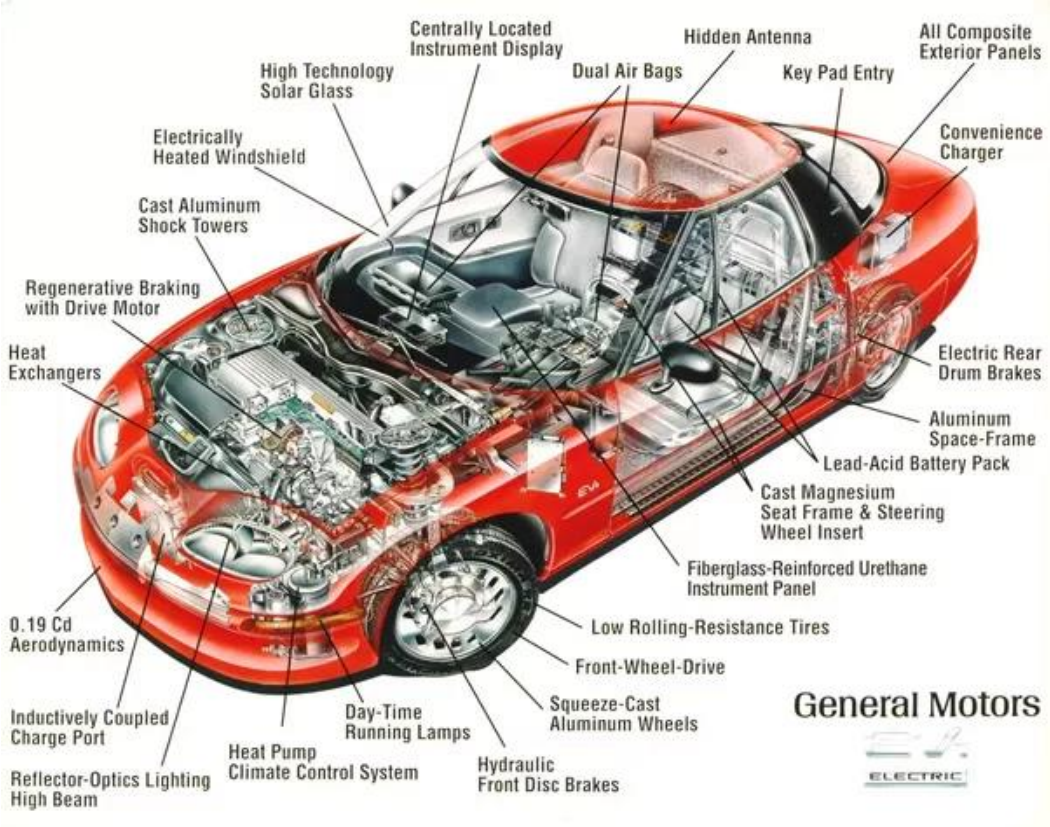
CS101: Introduction to Programming

Venkatesh Vinayakarao
Department of Computer Science and Engineering
IIIT Sri City, Chittoor.
venkatesh.v@iiits.in

Agenda

- Functions in C
  - Passing Values
  - Scope
  - Order of Evaluation
  - Return Type

Parts of a Car



- Car is made up of several parts!
- Similarly, parts of C program can be written and reused.

Example

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

int main()
{
    int num = 25;
    double squareRoot;

    squareRoot =  sqrt(num);
    printf("Square root of %d =  %lf", num, squareRoot);

    return 0;
}
```

Another Example

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

int main () {
    printf("Value 8 ^ 3 = %f\n", pow(8, 3));

    printf("Value 3 ^ 2 = %f", pow(3, 2));

    return(0);
}
```

What happens with %d?

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

int main () {
    printf("Value 8 ^ 3 = %d\n", pow(8, 3));

    printf("Value 3 ^ 2 = %d", pow(3, 2));

    return(0);
}
```

- Beware of return types.
- pow returns double!
- So, %d will convert it to zero.

Casting to int

```
1   #include <stdio.h>
2   #include <math.h>
3
4   int main () {
5       printf("Value 8 ^ 3 = %d\n", (int) pow(8, 3));
6
7       printf("Value 3 ^ 2 = %d", (int) pow(3, 2));
8
9       return(0);
10  }
```

Our Own Function

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

int main () {
    printf("Max of 3 and 2 is %d", max(3,2));
}

int max(int x, int y) {
    int result;
    if (x > y)
        result = x;
    else
        result = y;
    return result;
}
```

Write a Function to Add Three Numbers

```c
#include <stdio.h>

int main () {
    printf("5 + 3 + 2 = %d", add(5,3,2));
}

int add(int x, int y, int z) {
    return x + y + z;
}
```

Scope Rules

```c
#include <stdio.h>

int main () {
    int i = 10;
    printf("%d", check());
}

int check() {
    return i + 1;
}
```

```c
#include <stdio.h>

int main () {

    printf("%d", check());
}

int check() {
    int i = 10;
    return i + 1;
}
```

Order of passing arguments

```c
#include <stdio.h>

int main () {
    int i = 2;
    check(i, i++, ++i);
}

int check(int a, int b, int c) {
    printf("%d%d%d", a,b,c);
}
```

- Do not write such code.
- The order of evaluation is unspecified.
- Each compiler may give a different result.

Post/Pre Increment and Functions

```c
#include <stdio.h>

int main () {
    int i = 2;
    check(++i); //prints 3
    check(i++); //prints 3
    printf("%d", i); //prints 4
}

int check(int a) {
    printf("%d", a);
}
```

Same Function, Variable Arguments

```c
#include <stdio.h>

int main () {
    printf("%d", 1);
    printf("%d%d", 1, 2);
    printf("%d%d%d", 1, 2, 3);
    printf("%d%d%d%d", 1, 2, 3, 4);
}
```

How to Implement a Min Function?

```
int main()
{
    int count = 3;
    printf("sum = %d\n", sum(count, 1, 2, 3));
    count = 4;
    printf("sum = %d\n", sum(count, 3, 5, 10, 15));
    count = 5;
    printf("sum = %d\n", sum(count, 1, 2, 3, 4, 5));
    return 0;
}
```

Implementation of a Variadic Function

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

int sum(int num, ...)
{
    va_list valist;

    int sum = 0;

    va_start(valist, num);
    for (int i = 0; i < num; i++)
        sum += va_arg(valist, int);

    va_end(valist);

    return sum;
}
```

- Ellipsis (“…”) specifies that the function is variadic.
- va_list, va_start and va_end allow access to the arguments.
- First argument is fixed.

Beware of Return Types!

```c
#include <stdio.h>

int main () {
    sqr(3);
}
int sqr(int n) {
    return n * n;
}
```

- What happens when MAX_INT is passed?

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

int main () {
    sqr(INT_MAX);
}
int sqr(int n) {
    return n * n;
}
```

C:\ccode\fn8.exe

```
Process returned 0 (0x0)   execution time : 1.406 s
Press any key to continue.
```

Multiplication Table

```
int main()
{
    printf("%d * %d = %d\n", 1,1,1);
    printf("%d * %d = %d\n", 1,2,2);
    printf("%d * %d = %d\n", 1,3,3);
    printf("%d * %d = %d\n", 1,4,4);
    printf("%d * %d = %d\n", 1,5,5);
    printf("%d * %d = %d\n", 1,6,6);
    printf("%d * %d = %d\n", 1,7,7);
    printf("%d * %d = %d\n", 1,8,8);
    printf("%d * %d = %d\n", 1,9,9);
    printf("%d * %d = %d\n", 1,10,10);
    return 0;
}
```

Multiplication Table

```
int main()
{
    for (int i=1; i<=10; i++) {
        printf("%d * %d = %d\n", 1,i,i);
    }
    return 0;
}
```

Multiplication Table

```
int main()
{
    for (int i=1; i<=10; i++) {
        printf("%d * %d = %d\n", 1,i,i);
    }
    for (int i=1; i<=10; i++) {
        printf("%d * %d = %d\n", 2,i,2*i);
    }
    for (int i=1; i<=10; i++) {
        printf("%d * %d = %d\n", 3,i,3*i);
    }
    for (int i=1; i<=10; i++) {
        printf("%d * %d = %d\n", 4,i,4*i);
    }
    return 0;
}
```

Multiplication Table

```
#include <stdio.h>
int main()
{
    for (int j=1; j<=4; j++) {
        for (int i=1; i<=10; i++) {
            printf("%d * %d = %d\n", j,i,j*i);
        }
    }
    return 0;
}
```

Printing the Multiplication Table

```
#include <stdio.h>
int printRow(int x, int y) {
    printf("%d * %d = %d\n", x, y, x * y);
}
int printTable(int x) {
    for (int i=1; i<=10; i++) {
            printRow(x,i);
    }
}
int main()
{
    for (int j=1; j<=4; j++) {
        printTable(j);
    }
    return 0;
}
```

So, what did we discuss?

- Functions
- Scope Rule
- The Case of Multiplication Table

Questions?