# C programming language pt.4

**Explicit casting:**

It happens when you convert from a larger data type.

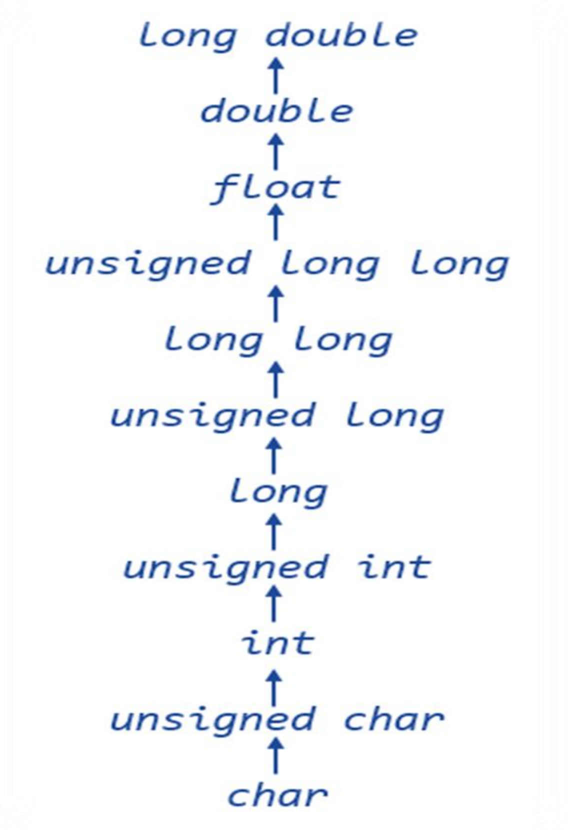
The compiler doesn't do it automatically, but you tell the compiler.

Data might be lost.

**Implicit casting:**

It happens when you convert from a smaller data type.

The compiler does it automatically when operations is applied.



```
result = (float)operant1 / (float)operant2;
```

It is recommended that you do all the casting explicitly (manually). Don't let the compiler do implicit casting.

## Memory layout:

- Stack segment.
- Heap segment.
- Data segment.
- Code segment.
- Block started by symbol (BSS) segment.

Each segment has own read, write and execute permissions.

In microcontrollers there are two main segments:

**Flash:**

keeps information even if power is off (nonvolatile).

.text/code section has your code or syntax.

.rodata (read only data) has your constant global variables.

.data has a copy of your initialized global variables (can't be changed).

**RAM:**

stack has your local variables. Either initialized or not or initialized to zero.

heap has your dynamic variables. (In runtime)

.bss has your non-initialized global variables or initialized to zero.

.data has your constant global variables.

***Startup code:***

*It copies .data in flash to .data in ram.*

*It makes variables in .bss equal zero.*

While calling a function the data in the function is created in stack (stack frame).

After that the stack frame is deleted from the stack area.

Constant variables in stack are marked to distinguish them.

## <u>Storage classes:</u>

When we create a variable there are two things attached to it:

1. Data type
2. Storage class

Storage class decides the <u>*(extent)*</u> lifetime and <u>*(visibility)*</u> scope of variables.

Storage class determines the scope to be global or local.

Storage class determines the lifetime to be static or automatic.

storage class determines the memory layout.

There are four types of scope:

1. Local (block)
2. Global (File)
3. Function

storage class:

1. Auto
2. Register
3. Extern
4. static

**auto:**

Every local variable is auto by default.

It is recommended to define every variable in the start of the function.

Only the block in which the auto variable is declared can access it.

**Register:**

Variables are stored in CPU instead of ram. This increases the speed.

Address of this variable can't be taken.

**Extern:**

It is used to declare functions and variables.

Extern variables can't be initialized.

Any static global variable or static function can be extern.

**static:**

static global variable or function is file scope.

Static local variable is created in .data in flash memory not stack.

When you include files in your main source file it should be header files only.

You shouldn't define anything in header files. you just put declaration.

You make getter and setter to access your static variables in another module.