CSE102
Computer Programming

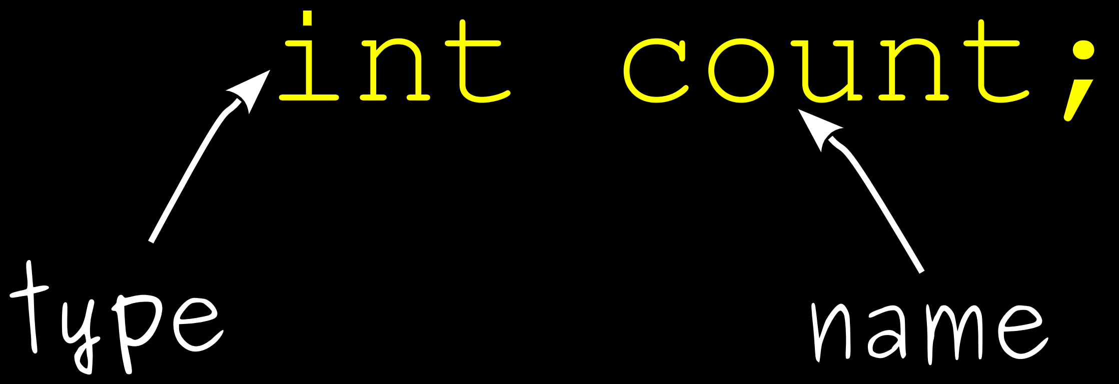
Credits: psychologywizard.net, livingwithra-nan.blogspot.in

Know Your Variables

Variables must have a name

Variables must have a type

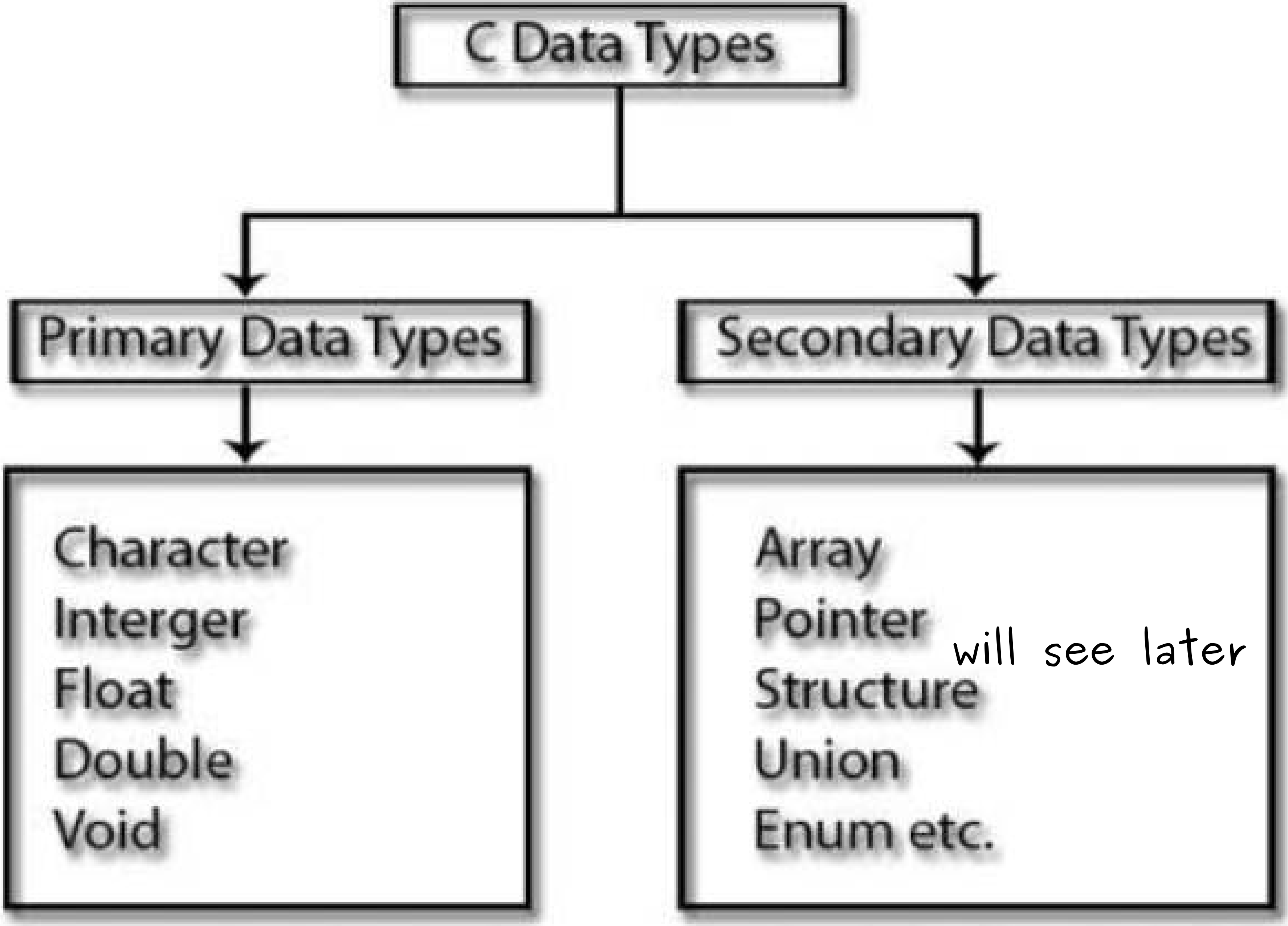
Variables must be declared before their usage



```
int count;
```

Assignment

```
  // ways to assign values to variables
x = 12;  // direct assignment of literal
         // value to variable
y = z;  // assign value of one variable
        // to another variable
z = x + 43;  // thru an expression
status = GetRadarInfo();
             // output of a function
scanf(“%d”, &num);// thru user input
```



C Data Types

Primary Data Types

- Character
- Interger
- Float
- Double
- Void

Secondary Data Types

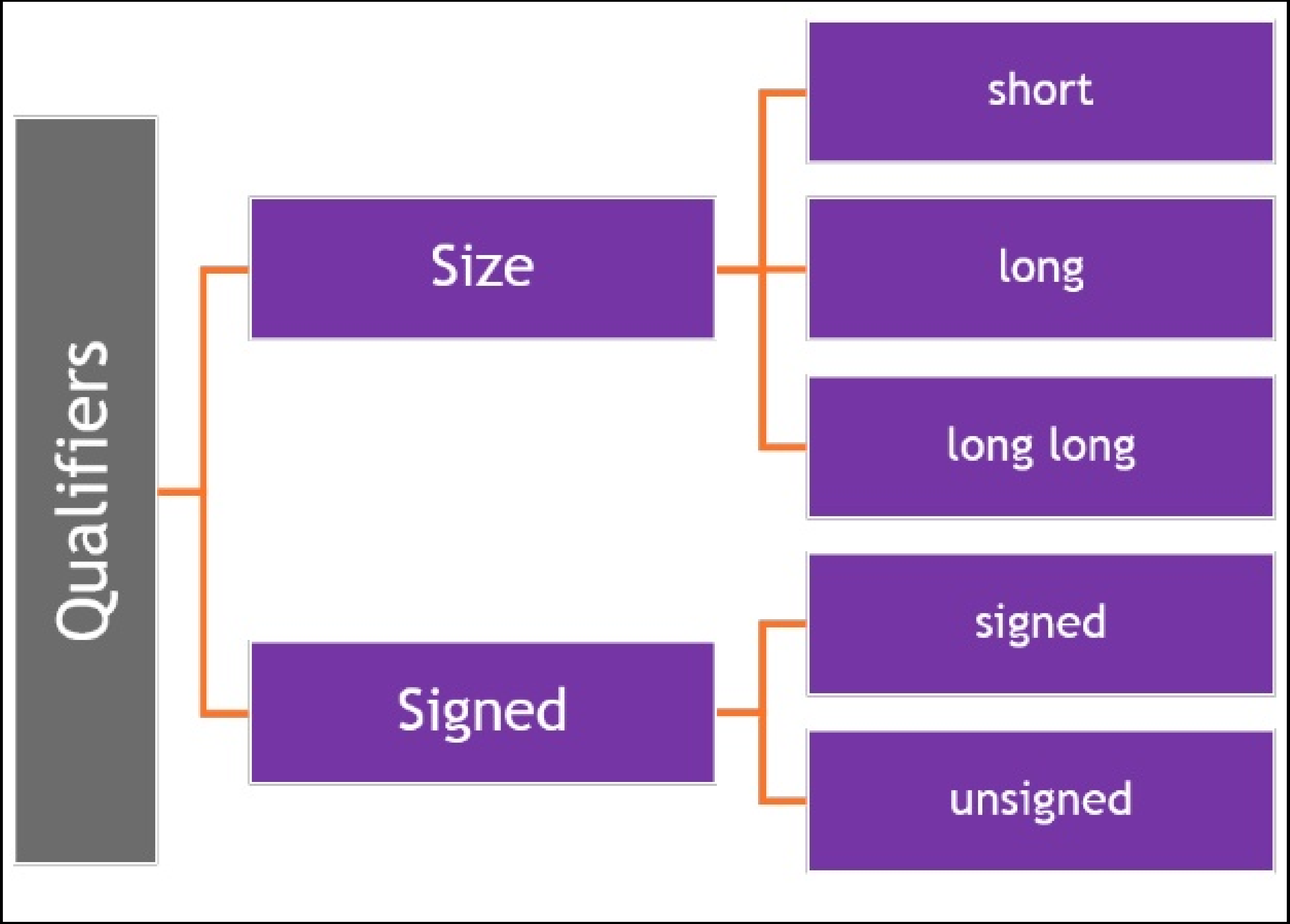
- Array
- Pointer
- Structure
- Union
- Enum etc.

will see later

void var; is not possible

Credits: www.eazynotes.com

Type & Sign Qualifiers



Qualifiers

Size

- short
- long
- long long

Signed

- signed
- unsigned

Credits: EMERTXE C Refresher

Negative Number Fuss

Signed Magnitude

```
00001010 (decimal 10)
10001010 (decimal -10)
```

Negative Number Fuss

Signed Magnitude

```
  00001010 (decimal 10)
+ 10001010 (decimal -10)
---------------------------
  10010100 (decimal -20)
```

Oops! signed magnitude does'nt support binary arithmetic!!

One’s Complement

Say Unsigned Integers

`00001010 (decimal 10)`

`11110101 (decimal 245)`

The complement of a number is the largest number represented with number of bits available minus the number itself

Observe: 255−10 = 245!!

One’s Complement

Interpreting it as negative counterpart!

00001010 (decimal 10)

+ 11110101 (decimal ~~245~~ -10)

11111111 (decimal ~~255~~ -0!)

Now if we claim let the MSB represent
Sign then 11110101 becomes -10!!
Alas! 11111111 now represents -0!!!
Binary arithmetic problem partially solved

One’s Complement

The Problem

```
  00000011 (decimal 3)
+ 11111101 (decimal -2)
-----------------------
  00000000 (decimal +0)
```

Remember 00000010 is 2 and its one’s Complement 11111101 is −2

wrong result? How do we fix it?

One’s Complement

The Fix

  00000011 (decimal 3)

+ 11111101 (decimal -2)

  00000000 (decimal +0)

+ 00000001 (the carry)

  00000001 (decimal 1)

Remember in the addition we had a 1 as carry!!! Just add it

One's Complement

A Cross Check

```
  00001010 (decimal 10)
+ 11111010 (decimal -5)
-----------------------
  00000100 (decimal 4)
+ 00000001 (the carry)
-----------------------
  00000101 (decimal 1)
```

It works!!!!

Used by many computers at one point in time like PDP-1 (DEC's 1st computer)

One's Complement

A Second Look

```
  00001010 (decimal 10)
+ 11110101 (decimal 245)
--------------------------
  11111111 (decimal 255)
```

Adding a number with its complement gives all ones (makes sense as 255−10=245)

Two’s Complement

Cause of More Fuss!!

```
  00001010 (decimal 10)
+ 11110101 (decimal 245)

  11111111 (decimal 255)
+ 00000001 (add 1)

  00000000 (decimal 0)
```

What if we add 1 to the addition?

Two's Complement

Cause of More Fuss!

00001010 (decimal 10)
+ 11110101 (decimal 245)

~~11111111 (decimal 255)~~
+ 00000001 (add 1)

00000000 (decimal 0)

Let's remove intermediate addition

Two's Complement

Cause of More Fuss

```
  00001010 (decimal 10)

  11110101 (decimal 245)
+ 00000001 (add 1)
-----------------------
  00000000 (decimal 0)
```

And focus on complement and adding one

Two's Complement

Demystifying

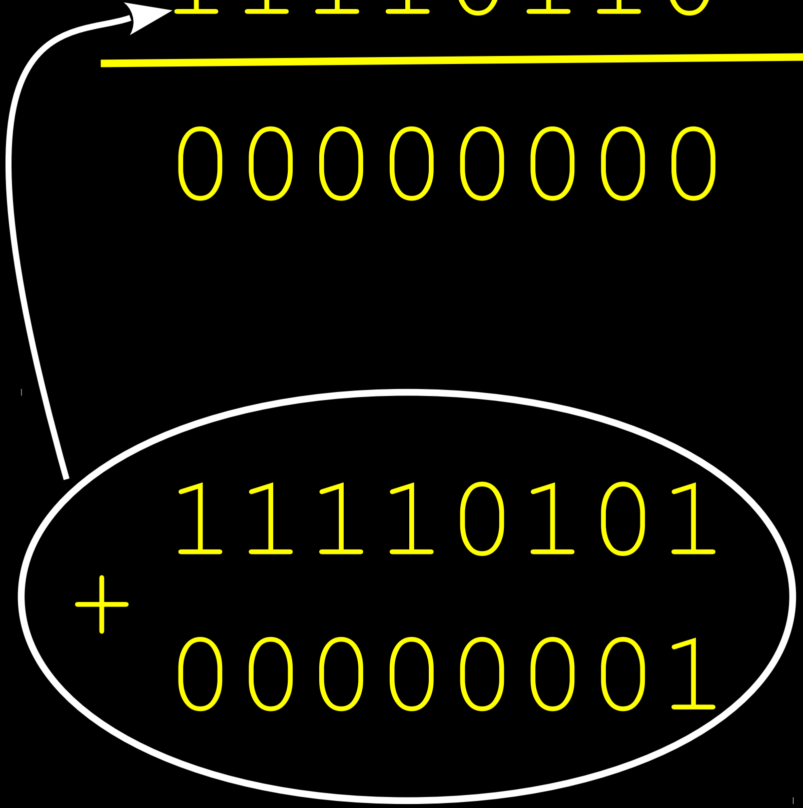
```
  00001010 (decimal 10)
  11110101 (decimal 245)
+ 00000001 (add 1)
-----------------------
  00000000 (decimal 0)
```

What if we combine complement and 1?

Two's Complement

Demystifying

```
  00001010 (decimal 10)
+ 11110110 (decimal ??)
-----------------------
  00000000 (decimal 0)
```



```
  11110101 (the complement)
+ 00000001 (add 1)
```

Two's Complement

Demystified

```
  00001010 (decimal 10)
+ 11110110 (decimal ??)
-------------------------
  00000000 (decimal 0)
```

What could be this number?
Which number when added to 10 will
give 0?

Two’s Complement

Tracing Our Path

```
  00001010 (decimal 10)


  11110101 (one’s complement)
+ 00000001 (add 1)
  ------------------------
  11110110 (decimal -10)
```

Two's Complement

Bidirectional!!

```
  11110110 (decimal -10)

  00001001 (one's complement)
+ 00000001 (add 1)
  
  00001010 (decimal 10)
```

One's Complement

Remember the Problem

$$\begin{array}{r l} & \texttt{00000011 (decimal 3)} \\ + & \texttt{11111101 (decimal -2)} \\ \hline & \texttt{00000000 (decimal +0)} \end{array}$$

What happens if we use two's complement?

Two’s Complement

Acid Test!!

00000011 (decimal 3)
+ 11111110 (decimal -2)

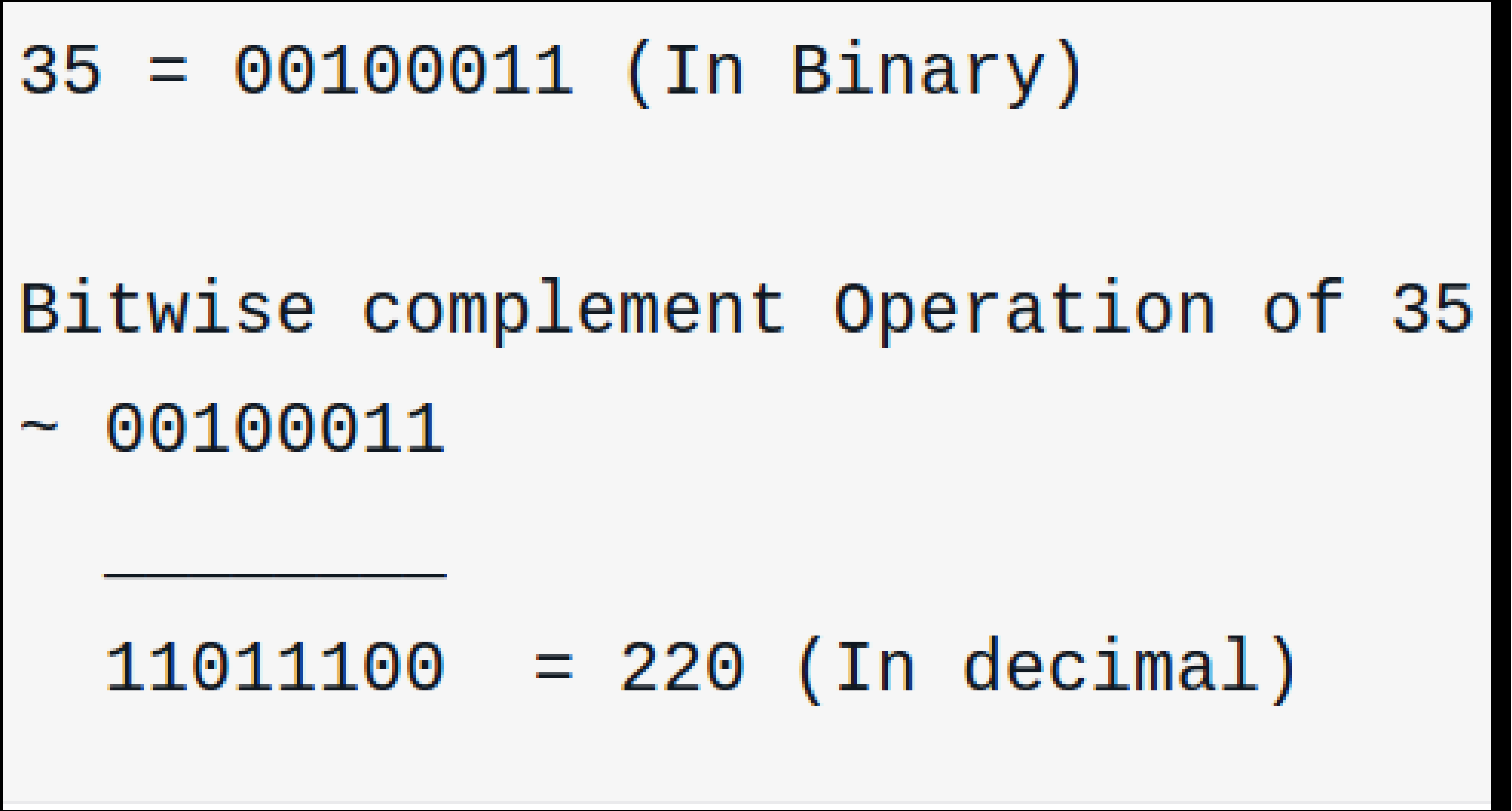
00000001 (decimal +1)

00000010 is decimal 2, 11111101 is one’s complement, 11111110 is two’s complement and that is −2

It works!!!!

Bitwise Complement ~

One Last Unfinished Business!!

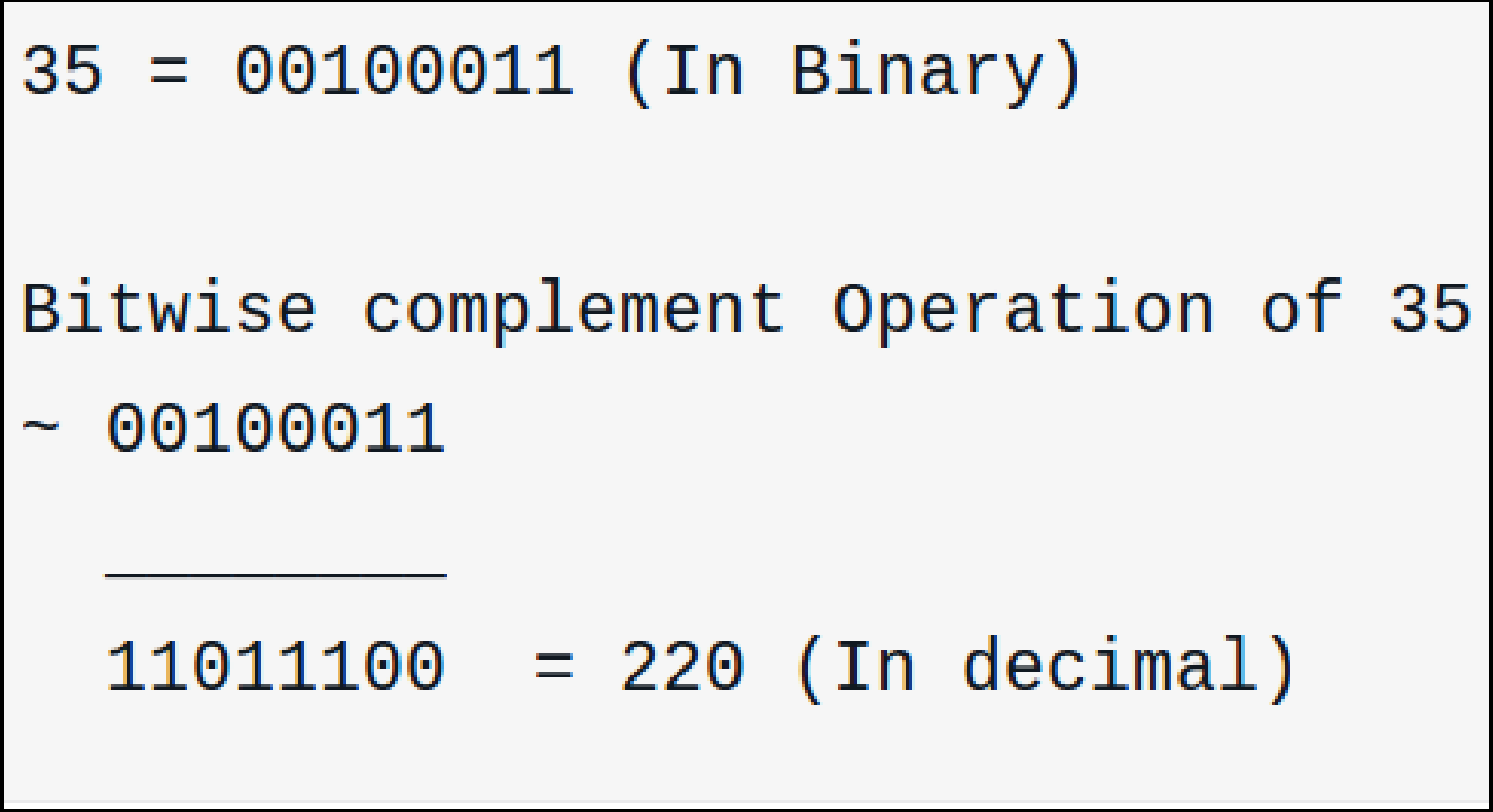


```
35 = 00100011 (In Binary)


Bitwise complement Operation of 35
~ 00100011
  ________
  11011100  = 220 (In decimal)
```

Credits: www.programiz.com

Bitwise Complement ~

How is 220 equivalent to -36?

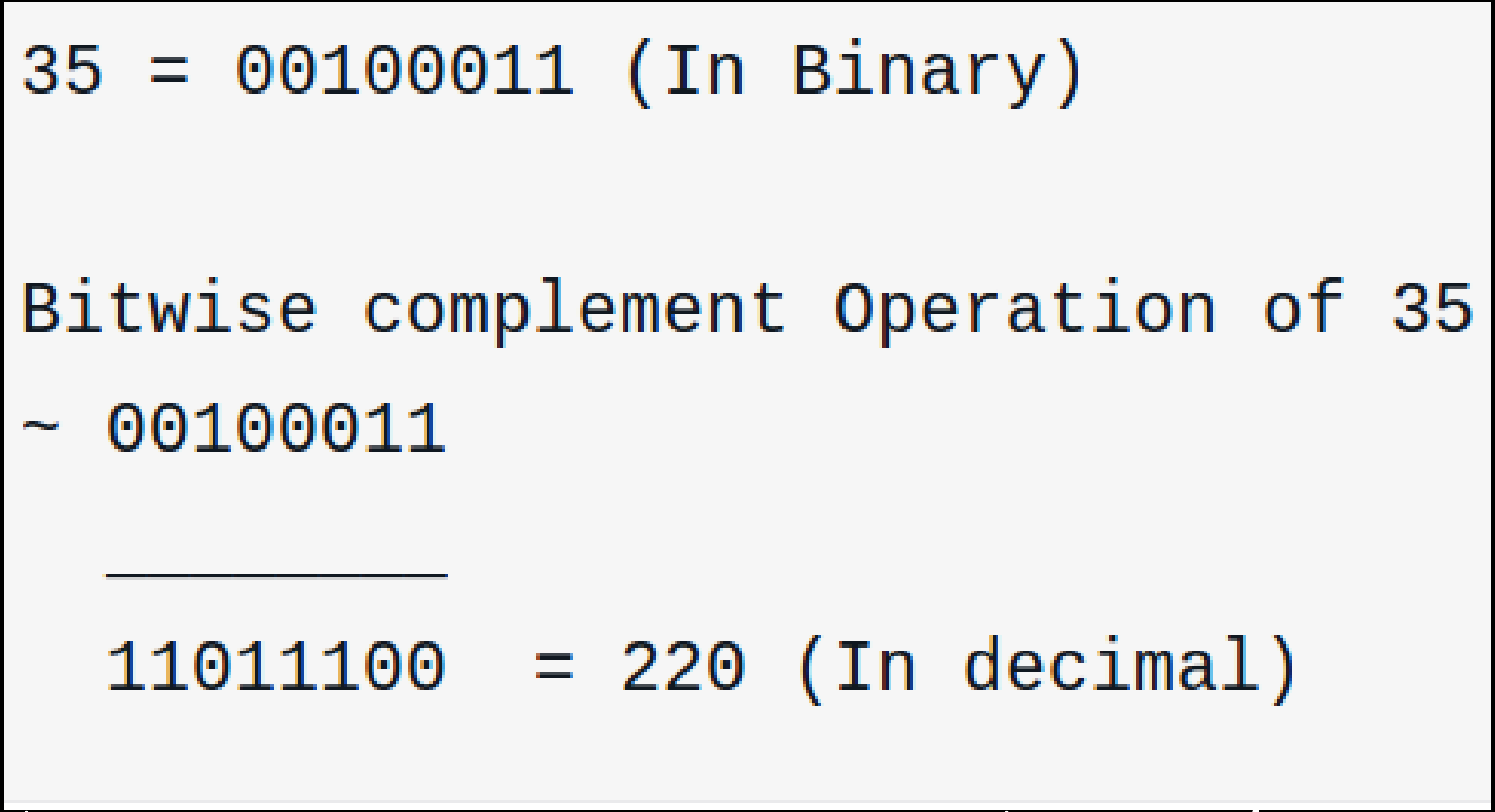


```
35 = 00100011 (In Binary)

Bitwise complement Operation of 35
~ 00100011

  ________
  11011100  = 220 (In decimal)
```

But the bitwise complement of
35 is −36 how?

Bitwise Complement ~

Should’nt we Check?



```
35 = 00100011 (In Binary)


Bitwise complement Operation of 35

~ 00100011

  ________

  11011100  = 220 (In decimal)
```

Negative numbers are stored as two’s complement of positive counterpart.

220 is two’s complement of −36!!

Two’s Complement

Is 220 equivalent to -36?

```
  1101110 (decimal 220)


  00100011 (one’s complement)
+ 00000001 (add 1)
  ______________________________
  00100100 (decimal 36)
```

Since two’s complement of a negative number gives it’s positive counterpart

1101110 must be decimal -36!!

Two’s Complement

Final Quantification

Since two’s complement of a negative number gives it’s positive counterpart and vice verse

`~(-x) + 1 = -(-x)!!`

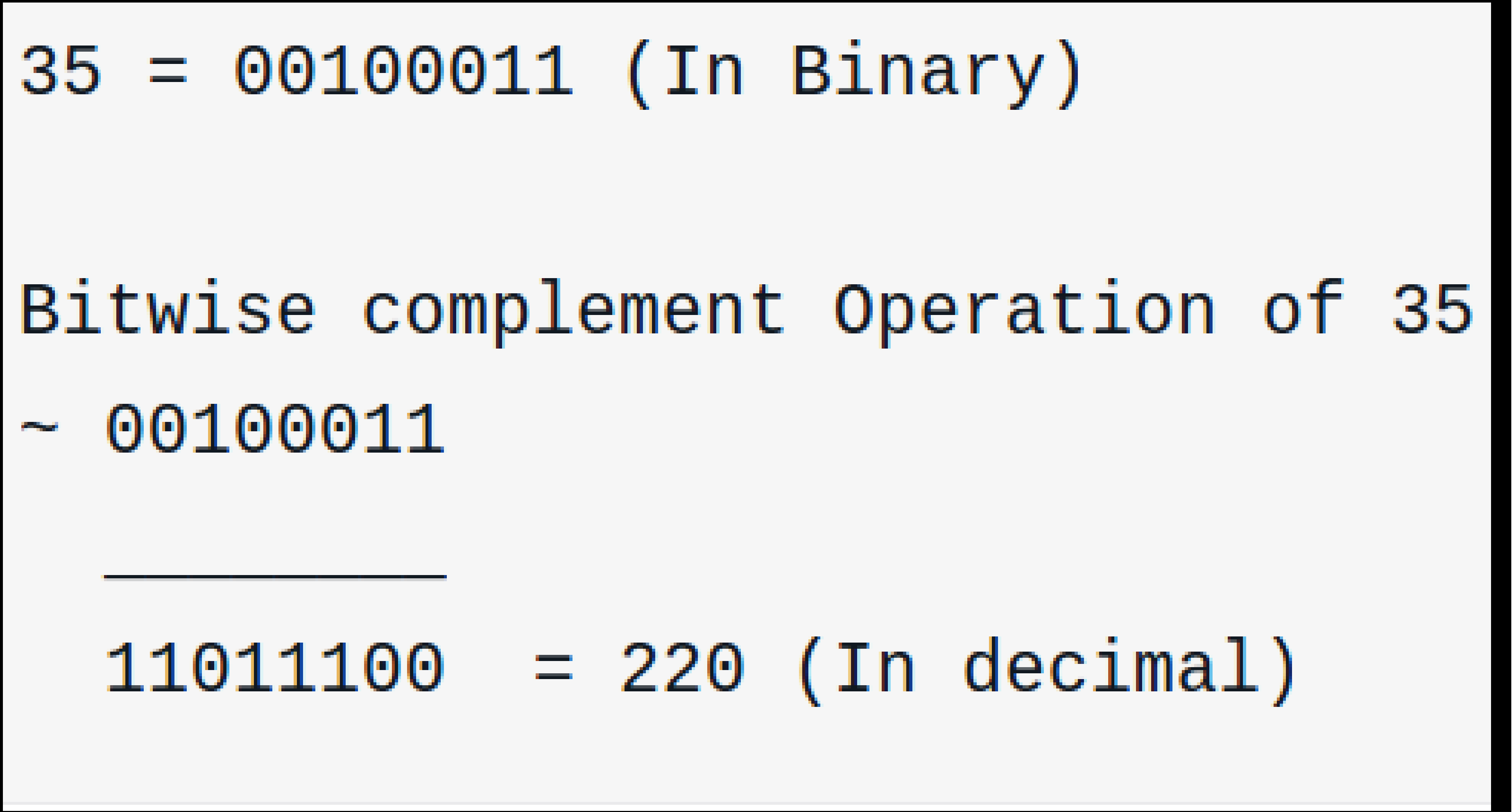
`~x + 1 = -x`

`~x = -x-1 (little arithmetic)`

`So ~35 = -35-1`

Bitwise Complement ~

~35 = -35-1 = -36



```
35 = 00100011 (In Binary)


Bitwise complement Operation of 35
~ 00100011
  ________
  11011100  = 220 (In decimal)
```

Credits: www.programiz.com

Constant & Volatile

```
// creates read-only variables
// value of pi can't be changed

const double pi = 3.141593;

// volatile variables can be changed
// by external agencies other than
// program

volatile int io_buf;

const volatile int io_buf;
```

Attributes of Variable

Name

Address

Type

Size

Value

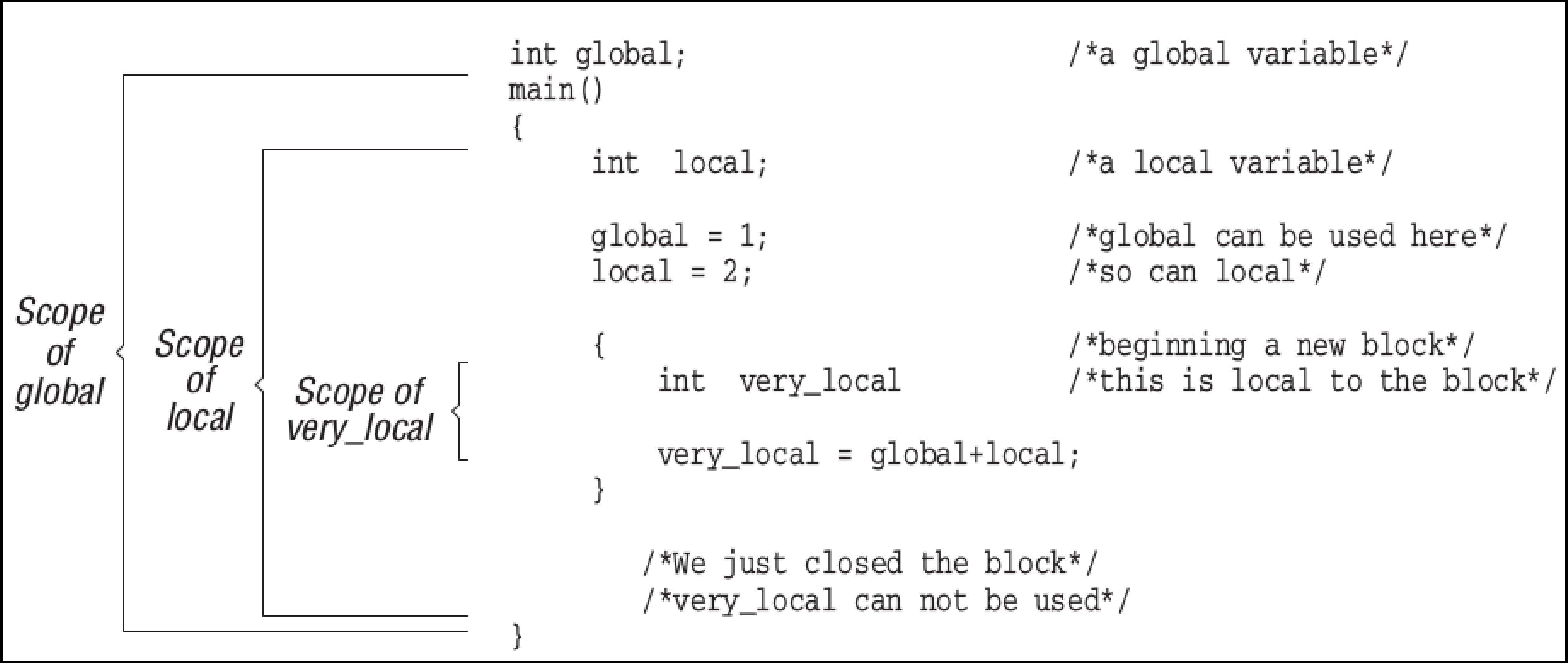
Storage Class
(Scope, visibility
and lifetime)

Credits: programiz.com

Local/Global Vars

Scope, Visibility and Lifetime

Notice variable declaration outside main()!!

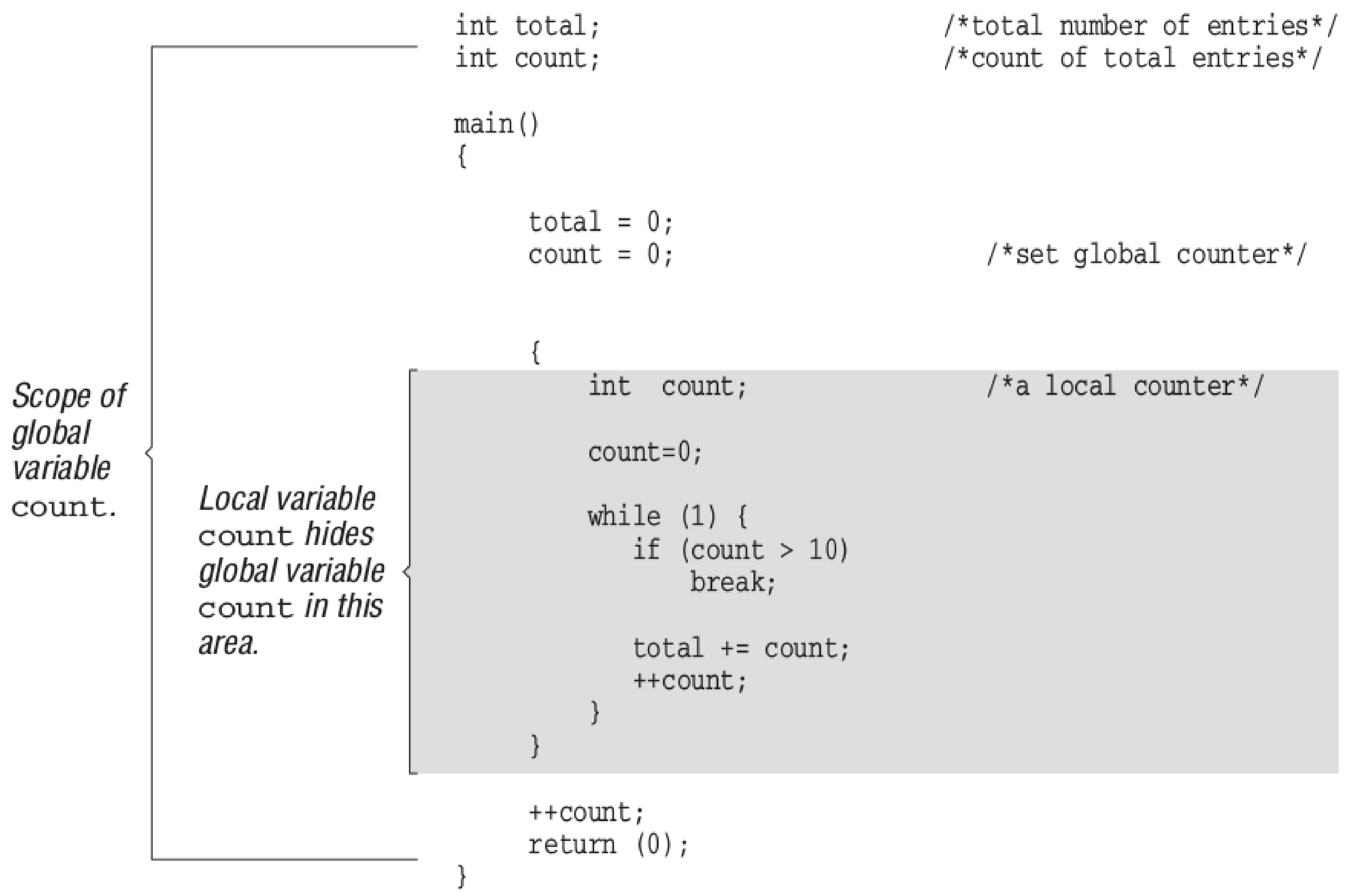


```
int global;                          /*a global variable*/
main()
{
     int  local;                     /*a local variable*/

     global = 1;                     /*global can be used here*/
     local = 2;                      /*so can local*/

     {                               /*beginning a new block*/
         int  very_local             /*this is local to the block*/

         very_local = global+local;
     }

        /*We just closed the block*/
        /*very_local can not be used*/
}
```

Credits: Practical C Programming

Local/Global Vars

Co-Existence!!!!



```
int total;                                 /*total number of entries*/
int count;                                 /*count of total entries*/

main()
{

     total = 0;
     count = 0;                               /*set global counter*/


     {
         int  count;                          /*a local counter*/

         count=0;

         while (1) {
            if (count > 10)
                break;

            total += count;
            ++count;
         }
     }

     ++count;
     return (0);
}
```

Scope of global variable `count`.

Local variable `count` *hides global variable* `count` *in this area.*

Credits: Practical C Programming

Storage Class

Automatic Variables

```
int sum_range(int lo, int hi){
  auto int i;
  auto int sum = 0;

  for(i=lo; i<=hi; i++){
    sum += i;
  }

  return sum;
}
```

Storage Class

Automatic Variables

```
int sum_range(int lo, int hi){
   auto int i;
   auto int sum = 0;

   for(i=lo; i<=hi; i++){
      sum += i;
   }

   return sum;
}
```

lo and hi are automatic too!

By default all local variables are automatic! So explicit auto qualifier is redundant

Storage Class

Static Variables

```
for(i=0; i<=5; i++){
  int n=0;
  printf(" %d ", ++n);
}
```

Predict the output!!

Storage Class

Static Variables

```
for(i=0; i<=5; i++){
   int n=0;
   printf(" %d ", ++n);
}
```

1 1 1 1 1 1

Storage Class

Static Variables

```
for(i=0; i<=5; i++){
   int n=0;
   printf(“ %d ”, ++n);
}
```

1 1 1 1 1 1

observe static qualifier

now what happens?

```
for(i=0; i<=5; i++){
   static int n=0;
   printf(“ %d ”, ++n);
}
```

Storage Class

Static Variables

```
for(i=0; i<=5; i++){
   int n=0;
   printf(“ %d ”, ++n);
}
```

```
1 1 1 1 1 1
```

```
for(i=0; i<=5; i++){
   static int n=0;
   printf(“ %d ”, ++n);
}
```

```
1 2 3 4 5 6
```

Storage Class

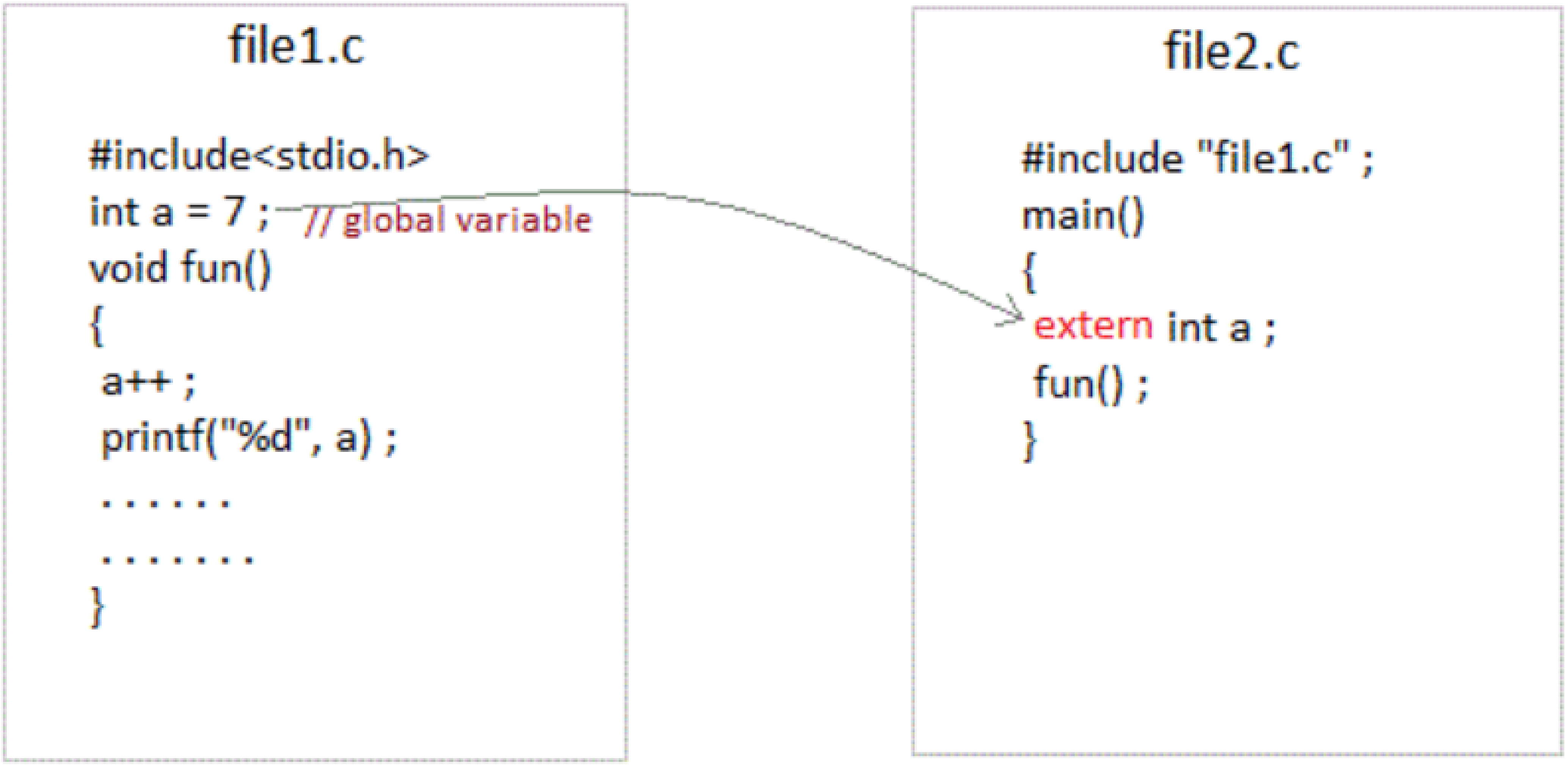
Register Variables

```
register int number;
```

register qualifier informs compiler to store variable in register instead of memory for faster access than normal variable

Storage Class

External Variables



```
#include<stdio.h>
int a = 7 ; // global variable
void fun()
{
 a++ ;
 printf("%d", a) ;
 . . . . . .
 . . . . . . .
}
```

```
#include "file1.c" ;
main()
{
 extern int a ;
 fun() ;
}
```

global variable from one file can be used in other using **extern** keyword.

Credits: www.studytonight.com

Predict the Ouput

```
int i;
int main(){
  i=0;
  i++;
  printf("Value of i is %d\n", i);
  func();
  printf("Value of i is %d\n", i);
}

void func(void){
  i++;
  i += 3;
}
```

Predict the Ouput

```
int i;
int main(){
  i=0;
  i++;
  printf(“Value of i is %d\n”, i);
  func();
  printf(“Value of i is %d\n”, i);
}

void func(void){
  i++;
  i += 3;
}
```

Value of i is 1
Value of i is 5

Predict the Ouput

```
int main(){
  i=0;
  i++;
  printf(“Value of i is %d\n”, i);
  func();
  printf(“Value of i is %d\n”, i);
}
int i;
void func(void){
  i++;
  i += 3;
}
```

Predict the Ouput

```
int main(){
   i=0;
   i++;
   printf(“Value of i is %d\n”, i);
   func();
   printf(“Value of i is %d\n”, i);
}
int i;
void func(void){
   i++;
   i += 3;
}
```

Compilation Error!!

Predict the Ouput

```
int main(){
   extern int i;
   i=0;
   i++;
   printf("Value of i is %d\n", i);
   func();
   printf("Value of i is %d\n", i);
}
int i;
void func(void){
   i++;
   i += 3;
}
```

Predict the Ouput

```
int main(){
   extern int i;
   i=0;
   i++;
   printf(“Value of i is %d\n”, i);
   func();
   printf(“Value of i is %d\n”, i);
}
int i;
void func(void){
   i++;
   i += 3;
}
```

Value of i is 1
Value of i is 5

Predict the Ouput

```
int main(){
  auto int i=1; {
    auto int i=2; {
      auto int i = 3;
      printf(“%d ”, i);
    }
    printf(“%d ”, i);
  }
  printf(“%d ”, i);
}
```

Predict the Ouput

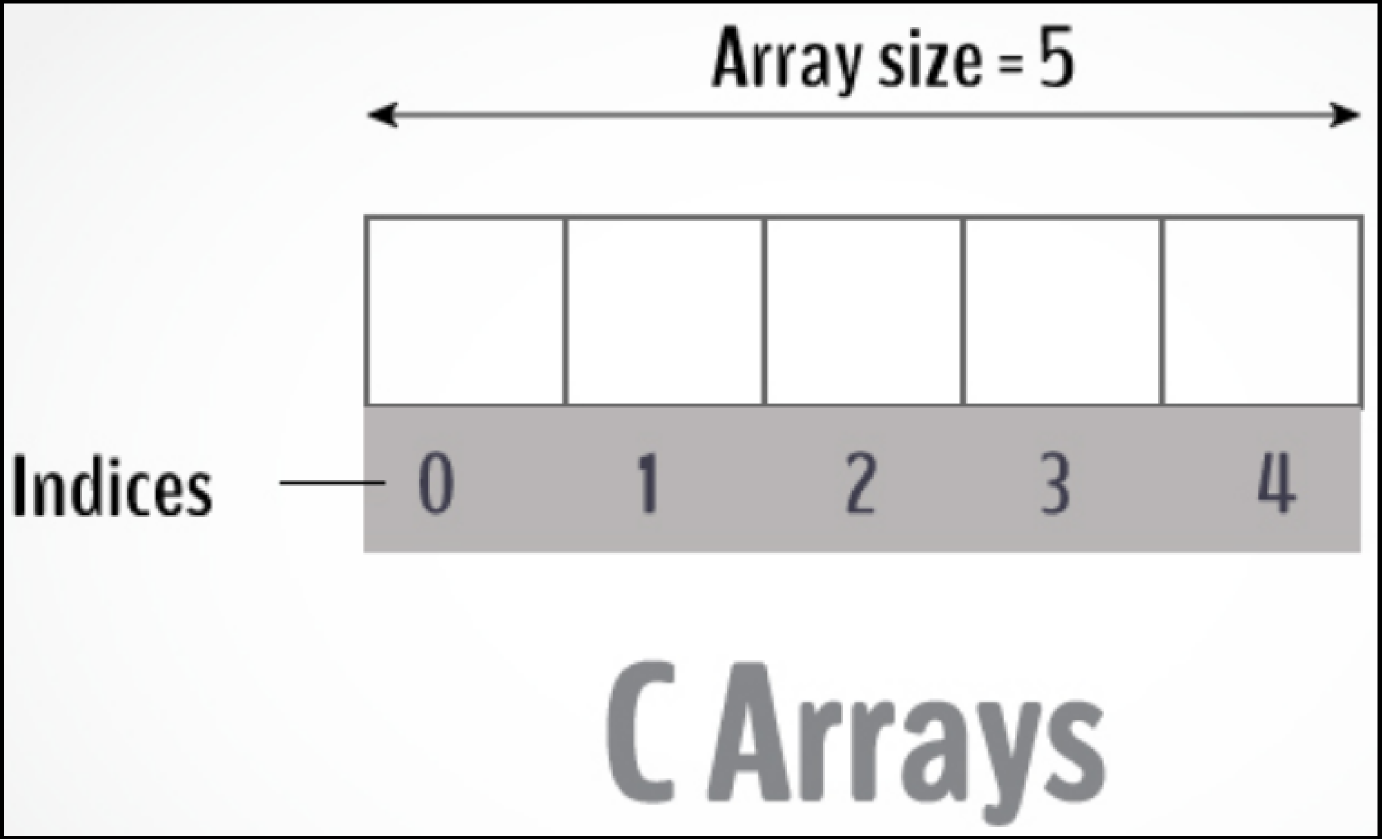
```
int main(){
 auto int i=1; {
  auto int i=2; {
   auto int i = 3;
   printf(“%d ”, i);
  }
  printf(“%d ”, i);
 }
 printf(“%d ”, i);
}
```

3 2 1

CSE102

Computer Programming

(Next Topic)



Credits: www.programiz.com