power = capacitive load * Voltage^2 × Frequeny

Problem: Suppose we developed a new, simpler processor that has 85% of the capacitive load of the more complex older processor. Further, assume that it can adjust voltage so that it can reduce voltage 15% compared to processor B, which results in a 15% shrink in frequency. What is the impact on dynamic power? old

New processor

$C_{new} = 0.85\ C_{old}$.

$V_{new} = (100-15)\%\ V_{old}$.

$= 85\%\ V_{old}$

$= 0.85\ V_{old}$.

$C_{new} = 0.85\ f_{old}$

Old processor

capacitive load = C_{old}

Voltage = V_{old}.

frequency = f_{old}.

$\text{power}_{new} = 0.85\ C_{old} \times (0.85\ V_{old})^2 \times 0.85\ f_{old}$.

$\text{power}_{old} = C_{old} \times V_{old}^2 \times f_{old}$.

$$\frac{\text{power}_{new}}{\text{power}_{old}} = \frac{0.85\ \cancel{C_{old}} \times (0.85\ \cancel{V_{old}})^2 \times 0.85\ \cancel{f_{old}}}{\cancel{C_{old}} \times \cancel{V_{old}^2} \times \cancel{f_{old}}}.$$

$= 0.85 \times (0.85)^2 \times 0.85$

$= (0.85)^4 = 0.52$.

Amdahl's Law

This states that the overall performance improvement gained by optimizing a single part of a system is limited by the fraction of time that the improved part is actually used.

$$\text{Execution time after improvement} = \frac{\text{Execution time affected by improvement}}{\text{Amount of Improvement}} + T_{\text{unaffected}}$$

40 program

$$\underset{50}{T_{Improved}} = \frac{T_{affected}\ 60}{\text{Improvement factor}} + T_{unaffected}$$

$c = a + b$	20 s	
$c = a - b$	20 s	
* $c = a \times b$	60 s	

~~100 s~~

50 s.

$\frac{100}{2} = 50s.$

$\boxed{\text{Multiplication}} \to 80$

100

Problem: Suppose a program runs in 100 seconds in a computer, with multiply operations responsible for 80 seconds? How much do I need to improve the speed of the multiplication if I want my program run 2 times faster.

$$T_{improved} = \frac{T_{affected}}{\text{Improvement factor}} + T_{unaffected}.$$

$$50 = \frac{80}{n} + 20$$

$$\therefore \frac{80}{n} = 50 - 20 = 30$$

$$\therefore n = \frac{80}{30} = 2.66$$

Multiply operations should 2.66 Times faster.

Problem: Suppose a program runs in 100 seconds in a computer, with multiply operations responsible for 80 seconds? How much do I need to improve the speed of the multiplication if I want my program run 5 times faster.

$$T_{improved} = \frac{T_{affected}}{\text{Improvement factor}} + T_{unaffected}$$

$$20 = \frac{80}{n} + 20$$

$$\therefore \frac{80}{n} = 0$$

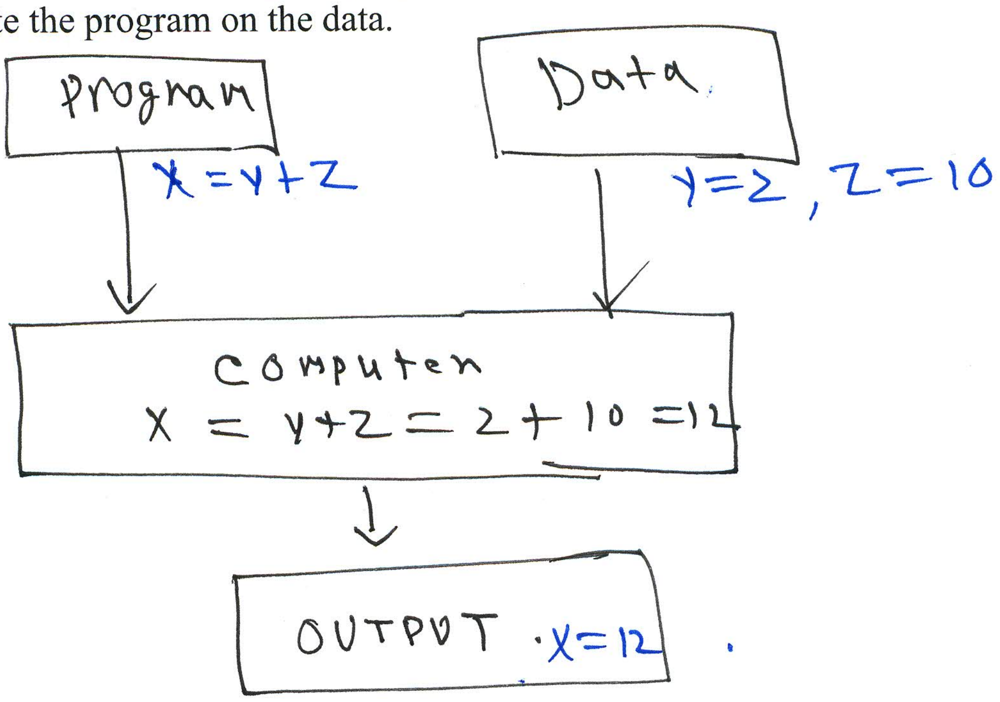
$$\therefore n = \frac{80}{0}$$

speed-up is not possible.

Review on C++

Running/Executing a Program

- The input to a computer can be thought of as consisting of two parts: a program and some data.
- Whenever we give a computer both a program to follow and some data for the program, we are said to be running the program on the data, and the computer is said to execute the program on the data.



Example 1: Suppose you want a computer to execute a simple instruction X = Y + Z to print the value of X on the output, where Y =2 and Z=10.

program OUTPUT. Data

Basics of C++/Analyzing Your First Program

```
# include <iostream>
using namespace std ;

int main()
{
   cout <<"Hello World"<< endl;
   return 0;
}
```

insertion

New Line.

OUTPUT.

```
Hello World.
```

include <iostream>

- Tells the computer to include a service for input/output.

```
using namespace std;
```

- Tells the computer to use the "standard namespace". Namespace is a mechanism for avoiding naming conflict in large programs.

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

- The construction defines a *function* called main that "returns" an "integer" (that is, a whole number without a fractional part, called int in C++) with value 0.

```
cout << "Hello World"<<endl;
```

- To display values on the screen, you use an entity called cout and the << operator (sometimes called the *insertion* operator).
- The endl symbol denotes an *end of line* marker. When this marker is sent to cout, the cursor is moved to the first column in the next screen row.

Modified First Program

What will be the output of the following program?

```
# include <iostream>
using namespace std ;

int main()
{
  → cout << "The answer is: " << 6 * 7<< endl;
    return 0;
}
```

OUTPUT

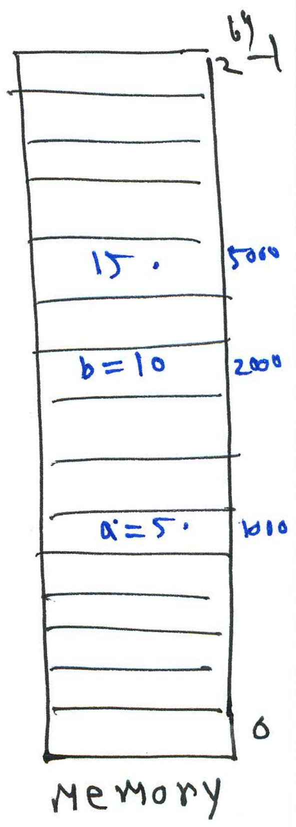
```
The answer is: 42 .
```

Variables

- A variable is a storage location in a computer program.
- Each Variable has a name and holds a value.

$a = 5$;
$b = 10$;

$c = a + b$
$= 5 + 10 = 15$



Number Type

- Numbers are two types: whole number and floating-point numbers

Whole number: 10 12 4 . int

Floating-point Number: 10.5 12.6
double .

Variable Names: Identifiers

- Variable names must start with a letter or underscore (_) character, the remaining characters must be letters, numbers, underscores.
- You can not use other symbols such as ? Or %. Spaces are not permitted.

x ✓ x,1 ✗ 12 ✗

x3 ✓ 3x ✗

myfirst;c ✗

- Variable names are case sensitive. Radius and radius are different names.

Rate rate RATE

- You can not use reserved words such as int, double, cout as names a = 3 ;

Variable Declarations

- Every variable in C++ program must be declared. When you declare a variable, you are telling the computer that what kinds of data you will be storing in the variable.
- When you declare more than one variables, the variables are separated by comma.

syntax .

```
Type_Name variable_Name1, variable_Name2,..;
```

```
int a ;
int a , b ;
```

$X+2 = Y+3$

Assignment Statement =

- An assignment statement place a new value into a variable.
- An **expression** is a number, a variable, or a combination of numbers and variables and operation symbols.

syntax .

```
variable = expression ;
```

```
a = 3 ;
```

a $\boxed{3}$

sum . number

$\boxed{\cancel{0}\ 5}$ $\boxed{5}$

```
sum = 0 ;
number = 5 ;
sum = sum + number ;
    = 0 + 5 = 5
```

C++

sum = sum + number

X = X + 2

Math

Y
$\boxed{\cdot}$

Uninitialized Variables

- A variable has no meaningful value until a program gives it one.
- For example, if the variable y has not been given a value, then the following assignment statement is an error.

```
→   int y ;      ← Variable Declaration
    y = y+2 ;    ← Assignment Statement
```

Variable Declaration With Initialization

You can initialize a variable at the time that you declare a variable.

```
Type_Name variable_Name1= Expression_for_value_1,
          variable_Name2= Expression_for_value_2,...;
```

```
int   y  = 10 ;
 y  =  y+2  ;
       10
```

Y
$\boxed{\cancel{10}}$
12 .