Tishk International University
Cybersecurity Department
Course Code: CBS 113

Programming Fundamentals

Lecture 3

Arithmetic, Casting, Random, Flowchart

Fall 2024

Hemin Ibrahim, PhD
hemin.ibrahim@tiu.edu.iq

Outline

- Arithmetic Operations
- Precedence (Priority) of Operations
- Mathematical Expressions
- Type Casting
- Character Literals
- Getline (String Object)
- Random Number
- Flowchart

Objectives

- Understand and apply basic arithmetic operations, including addition, subtraction, multiplication, division, and module.
- Comprehend the rules governing the order of execution in mathematical expressions.
- Importance of declaring variables and initializing them with values
- Understand the concept of type casting and its role in programming
- Understand the ASCII and Unicode encoding schemes for character representation.
- Explore the generation of random numbers in programming
- Learn the basic symbols and conventions used in flowchart diagrams

Arithmetic Operators

Operator	Meaning	Example
+	Addition	`total = cost + tax;`
-	Subtraction	`cost = total - tax;`
*	Multiplication	`tax = cost * rate;`
/	Division	`salePrice = original / 2;`
%	Modulus	`remainder = value % 3;`

Arithmetic Operators - Example

```cpp
#include <iostream>
using namespace std;
int main() {
    int a = 10, b = 3;
     cout << "Addition: " << a + b << endl;   // Output: 13
     cout << "Multi: " << a * b << endl;   // Output: 30

     return 0;
}
```

Arithmetic Operators - Example

```cpp
#include <iostream>
using namespace std;
int main() {
    double price = 19.99;
    int quantity = 3;

    double total = price * quantity;
    cout << "Total: " << total;

    return 0;
}
```

Output:

Total: 59.97

Arithmetic Operators

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

int main() {
    double basePayRate = 10.25;
    double regularHours = 40.0;
    double overtimePayRate = 20.5;
    double overtimeHours = 13;

    double regularWages, overtimeWages, totalWages;

    // Calculate the regular wages.
    regularWages = basePayRate * regularHours;

    // Calculate the overtime wages.
    overtimeWages = overtimePayRate * overtimeHours;

    // Calculate the total wages.
    totalWages = regularWages + overtimeWages;

    // Display the total wages.
    cout << "Wages for this week are $" << totalWages << endl;

    return 0;
}
```

Output:

Wages for this week are $676.5

Arithmetic Operators - Example

Create a C++ program to find the sale price of an item initially priced at $39.99, with a 20% discount. Output the **regular price**, **discount amount**, and **final sale price**.

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

int main() {
    // Variables to hold the regular price, the amount of a discount, and
        the sale price.
    double regularPrice = 39.99;
    double discount;
    double salePrice;

    // Calculate the amount of a 20% discount.
    discount = regularPrice * 0.2;

    // Calculate the sale price by subtracting the discount from the
        regular price.
    salePrice = regularPrice - discount;

    // Display the results.
    cout << "Regular price: $" << regularPrice << endl;
    cout << "Discount amount: $" << discount << endl;
    cout << "Sale price: $" << salePrice << endl;

    return 0;
}
```

Output:

Regular price: $39.99
Discount amount: $7.998
Sale price: $31.992

Precedence(Priority) of Operators

Order of operations was invented in 1912

People in **2023** :

$$6 \div 2(1+2) =$$

Precedence(Priority) of Operators

Precedence of Arithmetic Operators:

- Parentheses () are evaluated first. The expression in the innermost parentheses is evaluated first if the parentheses are nested.
- After parentheses multiplication (*), division (/), modulus (%) operators are evaluated.
- Addition (+) and Subtraction (-) are evaluated last.
- The operators with the same precedence are evaluated left to right.

```
3 * 5 + 2    = 17
3 * (5 + 2)  = 21
5 + 3 * 4 – 2 = 15
6 * 8 / 4    = 12
6 * (8 / 4)  = 12
```

```
(12 / (8 – 2)) = 2
8 + (7 - 9)    = 6
9 + 3 + 4 – 2  = 14
16 / 4 * 2     = 8
4 / 2 - 2      = 0
```

C++ Operators can be divided into 3 levels according to their precedence

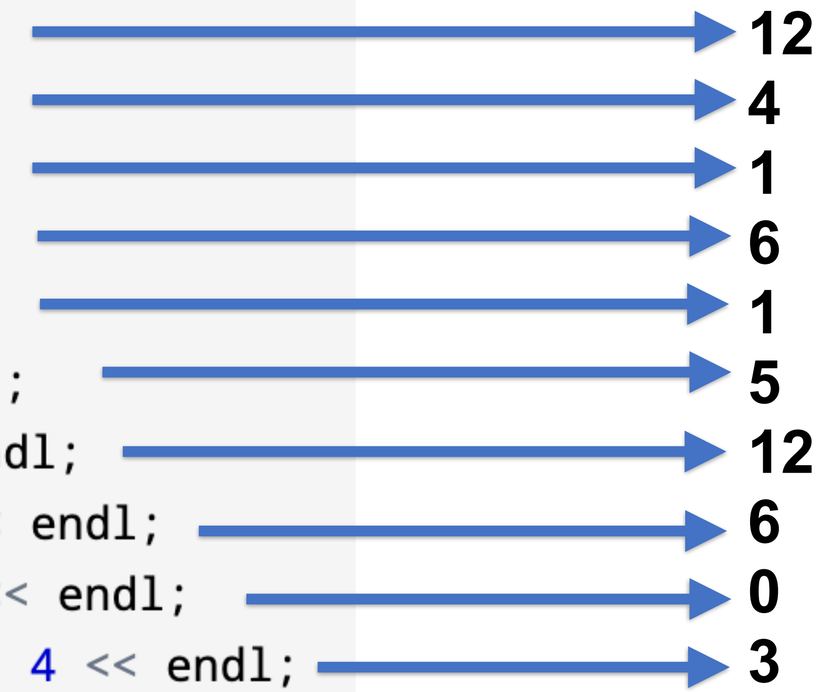
- First: **()**
- Second: ***** , **/** , **%**
- Third: **+** , **-**

Precedence(Priority) of Operators - Example

```cpp
#include <iostream>
using namespace std;
int main(){

    cout << 4 * 6 / 2 << endl;
    cout << 6 / 3 * 2 << endl;
    cout << 9 / 3 / 2 << endl;
    cout << 3 + 5 - 2 << endl;
    cout << 9 - 6 - 2 << '\n';
    cout << 9 - (6 - 2) << '\n';
    cout << 9 / 3 + 3 * 3 << endl;
    cout << 12 / (3 + 3) * 3 << endl;
    cout << 3 * 2 / 2 + 2 - 5 << endl;
    cout << 3 * 2 / 2 + 2 - 8 / 4 << endl;

    return 0;
}
```



Arithmetic expressions in C++

Arithmetic expressions in C++ must be entered into the computer in **straight line form.**

$$\frac{A + B}{C + D}$$ becomes in C++ `( A + B ) / ( C + D )`

$$A + \frac{B}{C} + D$$ becomes in C++ `A + ( B / C ) + D`

$$\frac{1}{1+2x^2}$$ becomes in C++ `1 / (1 + 2 * x * x)`

Arithmetic expressions in C++ Example

$$\frac{A + B}{C + D}$$

```cpp
#include <iostream>
using namespace std;
int main() {

    int A = 10, B = 5, C = 4, D = 2;
    int result = (A + B) / (C + D);
    cout << "The result is: " << result << endl;

    return 0;
}
```

Mathematical Expressions - Division

If one (or both) of the operands of the division operator is (are) double result will be double.

```
#include <iostream>
using namespace std;
int main(){
    double numerator, denominator;

    cout << "Enter the numerator: ";
    cin >> numerator;
    cout << "Enter the denominator: ";
    cin >> denominator;

    cout << "The decimal value is ";
    cout << (numerator / denominator) << endl;

    return 0;
}
```

```
Enter the numerator: 5
Enter the denominator: 2
The decimal value is 2.5
```

Mathematical Expressions - Division

If the operands of the division operator are both integers result will be integer.

```
#include <iostream>
using namespace std;
int main(){
    int numerator, denominator;

    cout << "Enter the numerator: ";
    cin >> numerator;
    cout << "Enter the denominator: ";
    cin >> denominator;

    cout << "The decimal value is ";
    cout << (numerator / denominator) << endl;

    return 0;
}
```

```
Enter the numerator: 5
Enter the denominator: 2
The decimal value is 2
```

Integer Division:

```cpp
#include <iostream>
using namespace std;
int main() {

    int x1 = 10, x2 = 3;
    int result = x1 / x2;
    cout << "Result (int / int): " << result << endl;
    // Output: Result (int / int): 3

    return 0;
}
```

Mixed Division with Double Result:

```cpp
#include <iostream>
using namespace std;
int main() {

    int x1 = 10, x2 = 3;
    double result = x1 / x2;
    cout << "Result (int / int, double result): " << result << endl;

    // Output: Result (int / int, double result): 3.0
    return 0;
}
```

Double Division with Integer Result:

```cpp
#include <iostream>
using namespace std;
int main() {

    double x1 = 10.0, x2 = 3.0;
    int result = x1 / x2;
    cout << "Result (double / double, int result): " << result << endl;

    // Output: (double / double, int result): 3
    return 0;
}
```

Double Division with Double Result

```cpp
#include <iostream>
using namespace std;
int main() {

    double x1 = 10.0, x2 = 3.0;
    double result = x1 / x2;
    cout << "Result (double / double): " << result << endl;

    // Output: (double / double): 3.33333
    return 0;
}
```

Type Casting

Used to convert one data type to another.

The general format of a type cast expression is:

Data_type(Value)

OR

(DataType)Value

Example:

double number = 3.7;

int val;

val = int(number);

```
#include <iostream>
using namespace std;
int main(){

    double number = 3.7;
    cout<<"number = "<<number<<endl;
    cout<<"int(number) = "<<int(number)<<endl;
    return 0;
}
```

Output:

```
number = 3.7
int(number) = 3
```

Type Casting

A value in any of the built-in types can be converted to any of the other types.

Example:

- (int) 3.14 // changes to int to give 3
- (double) 2 // changes 2 to a double type 2.0
- (char) 65 // change 65 to character A

Type Casting

Create a C++ program that asks the user for their reading habits. Prompt them to input the total number of books they plan to read and the months they'll spend reading. Calculate and show the average number of books per month.

```cpp
#include <iostream>
using namespace std;
int main(){

    int books; // Number of books to read
    int months; // Number of months spent reading
    double perMonth; // Average number of books per month

    cout << "How many books do you plan to read? ";
    cin >> books;
    cout << "How many months will it take you to read them? ";
    cin >> months;

    perMonth = double(books) / months;
    // Also you can write: perMonth = (double) books / months;

    cout << "That is " << perMonth << " books per month.\n";

    return 0;
}
```

Output:

```
How many books do you plan to read? 3
How many months will it take you to read them? 4
That is 0.75 books per month.
```

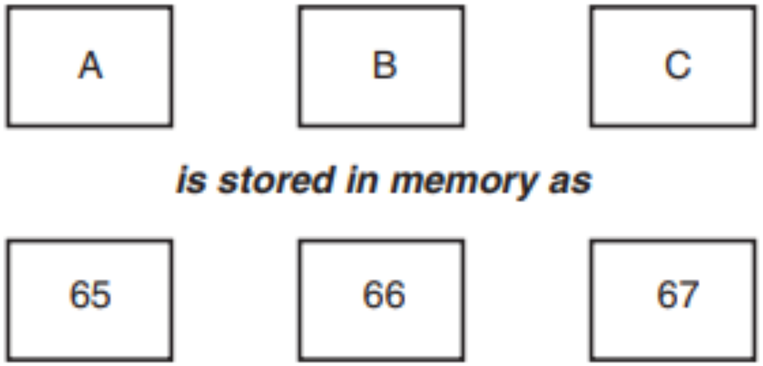
Arithmetic operators and expressions

```
16
17  cout << "10 + 3= "<< 10 + 3 << endl;
18  cout << "10 - 3= "<<10 - 3 << endl;
19  cout << "10 * 3= "<< 10 * 3 << endl;
20  cout << "11 / 5= "<<11 / 5 << "\n";
21  cout << "11.0 / 5.0= "<<11.0 / 5.0 << '\n';
22  cout << "(int)11 / 5= "<<(int)11 / 5 << '\n';
23  cout << "(int)11.0 / 5.0= "<<(int)11.0 / 5.0 << '\n';
24  cout << "(int)(11.0 / 5.0)= "<<(int)(11.0 / 5.0) << '\n'; //casting
25  cout << "(float)11 / 5= "<<(float)11 / 5 << "\n"; //casting
26  cout << "10.0 / 3= "<<10.0 / 3 << '\n';
27  cout << "10 / 3.0= "<<10 / 3.0 << '\n';
28  cout << "10 % 3= "<<10 % 3 << '\n';
29  cout << 5 << '+' << 1 << '=' << 5 + 1 << endl;
30
```

```
10 + 3= 13
10 - 3= 7
10 * 3= 30
11 / 5= 2
11.0 / 5.0= 2.2
(int)11 / 5= 2
(int)11.0 / 5.0= 2.2
(int)(11.0 / 5.0)= 2
(float)11 / 5= 2.2
10.0 / 3= 3.33333
10 / 3.0= 3.33333
10 % 3= 1
5+1=6
```

Character Literals

The most commonly used method for encoding characters is **ASCII**, which stands for the American Standard Code for Information Interchange.



ASCII Value	Char	ASCII Value	Char	ASCII Value	Char	ASCII Value	Cha
32	' '	61	=	81	Q	105	i
33	!	62	>	82	R	106	j
34	"	65	A	83	S	107	k
42	*	66	B	84	T	108	l
43	1	67	C	85	U	109	m
45	-	68	D	86	V	110	n
47	/	69	E	87	W	111	o
48	0	70	F	88	X	112	p
49	1	71	G	89	Y	113	q
50	2	72	H	90	Z	114	r
51	3	73	I	97	a	115	s
52	4	74	J	98	b	116	t
53	5	75	K	99	c	117	u
54	6	76	L	100	d	118	v
55	7	77	M	101	e	119	w
56	8	78	N	102	f	120	x
57	9	79	O	103	g	121	y
60	<	80	P	104	h	122	z

Char example

Write a program that is printing the corresponding ASCII of each letter of "TIU"

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

int main() {

    cout << "ASCII value of 'T': " << int('T') << endl;
    cout << "ASCII value of 'I': " << int('I') << endl;
    cout << "ASCII value of 'U': " << int('U') << endl;

    return 0;
}
```

Output:

```
ASCII value of 'T': 84
ASCII value of 'I': 73
ASCII value of 'U': 85
```

Characters and string Object

- Using **cin** with the **>>** operator for string input can lead to potential issues.
- **cin** ignores leading whitespace characters (**spaces**, **tabs**, **line breaks**) when reading input.
- Once **cin** encounters the first nonblank character, it **stops** reading at the next whitespace character.
- To address this limitation, the **getline** function in C++ can be used.
- **getline** reads an entire line, including leading and embedded spaces, and stores it in a string object..

Characters and string Object - Example

```cpp
#include <iostream>
using namespace std;
int main(){

    string name, city;

    cout << "Please enter your name: ";
    cin >> name;
    cout << "Enter the city you live in: ";
    cin >> city;

    cout << "Hello, " << name << endl;
    cout << "You live in " << city << endl;

    return 0;
}
```

Using cin>>

?

```
Please enter your name: Alan Ahmed
Enter the city you live in: Hello, Alan
You live in Ahmed
```

Characters and string Object - Example

```
#include <iostream>
#include <string>
using namespace std;
int main(){
    string name, city;

    cout << "Please enter your name: ";
    getline(cin, name);
    cout << "Enter the city you live in: ";
    getline(cin, city);

    cout << "Hello, " << name << endl;
    cout << "You live in " << city << endl;

    return 0;
}
```

Using getline

Adding <string>

Output:

```
Please enter your name: Alan Ahmed
Enter the city you live in: Erbil
Hello, Alan Ahmed
You live in Erbil
```

Random Numbers

- Random numbers are useful for lots of different programming tasks. The C++ library has a function, **rand()**, that you can use to generate random numbers.

```
#include <iostream>
using namespace std;
int main(){

    // Seed the random number generator.
    srand(time(0));

    // Display three random numbers.
    cout << rand() << endl;
    cout << rand() << endl;
    cout << rand() << endl;

    return 0;
}
```

Output:

```
602620988
1366704749
631888070
```

Random Numbers - Example

Generate random numbers in a specific ranges

```cpp
#include <iostream>
using namespace std;
int main() {

    srand(time(0));  // Seed the random number generator
    int randomNum = rand() % 100;  // Generate a number between 0 and 99
    cout << randomNum;

    return 0;
}
```

Random Numbers - Example

Generate random numbers in a specific ranges

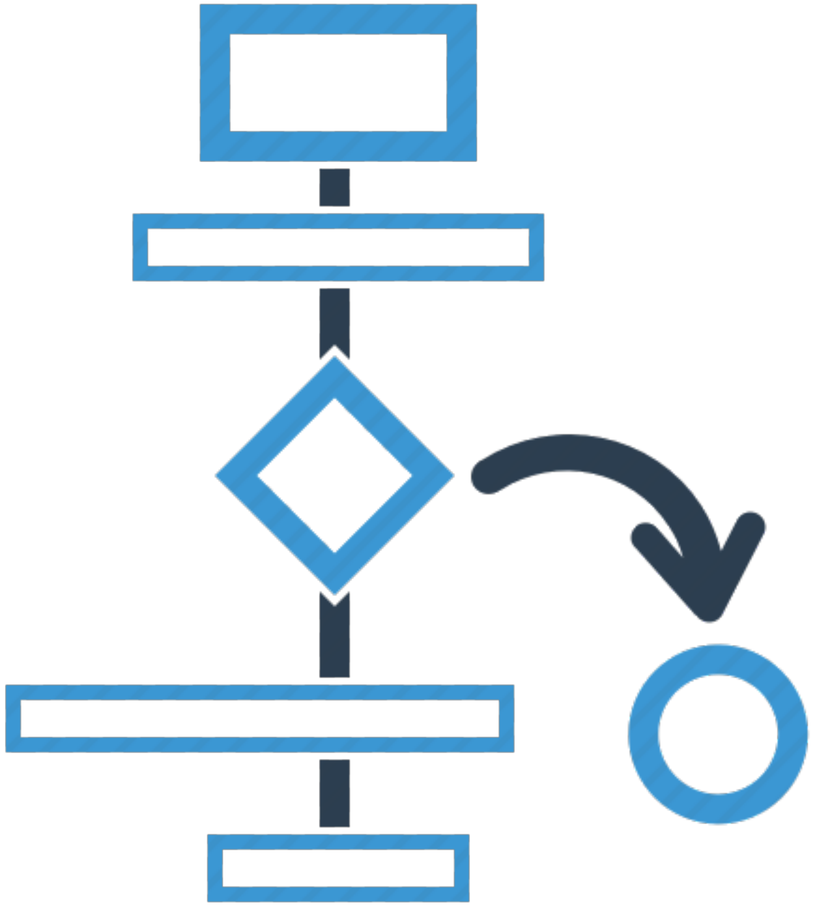
```
#include <iostream>
using namespace std;
int main() {

    srand(time(0));  // Seed the random number generator
    cout << "Random number (0-99): " << rand() % 100 << "\n";
    cout << "Random number (1-100): " << (rand() % 100) + 1 << "\n";
    cout << "Random number (20-50): " << (rand() % 31) + 20 << "\n";

    return 0;
}
```

Flowchart

- A flowchart is a picture (graphical representation) of the problem-solving process.
- A flowchart gives a step-by-step procedure for solution of a problem.
- Using flowcharts can show the sequence and logic of each step before writing a computer program.
- Even people with little programming knowledge can understand flowcharts.



Flowchart Elements

Ellipse

Rectangle

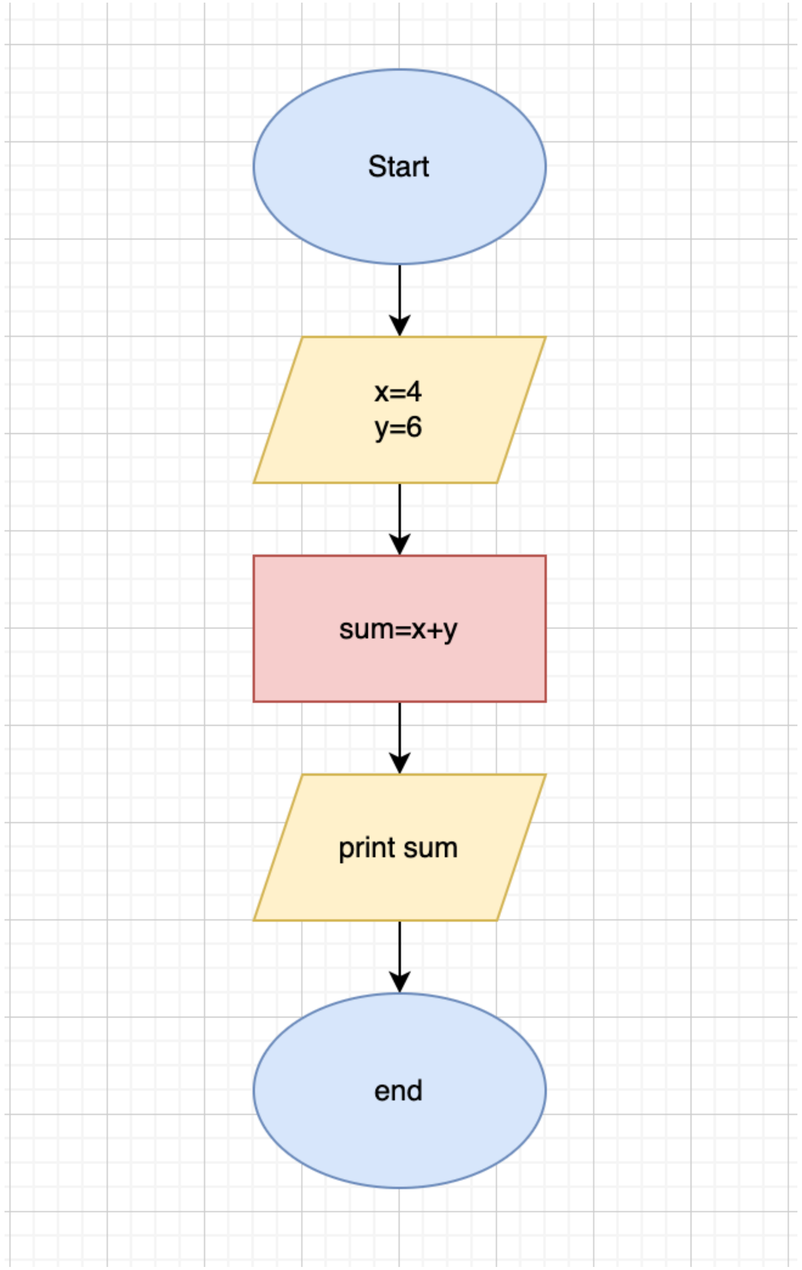
Parallelogram

Diamond

Symbol	Symbol Name	Purpose
	Start/Stop	Used at the beginning and end of the algorithm to show start and end of the program.
	Process	Indicates processes like mathematical operations.
	Input/ Output	Used for denoting program inputs and outputs.
	Decision	Stands for decision statements in a program, where answer is usually Yes or No.
	Arrow	Shows relationships between different shapes.

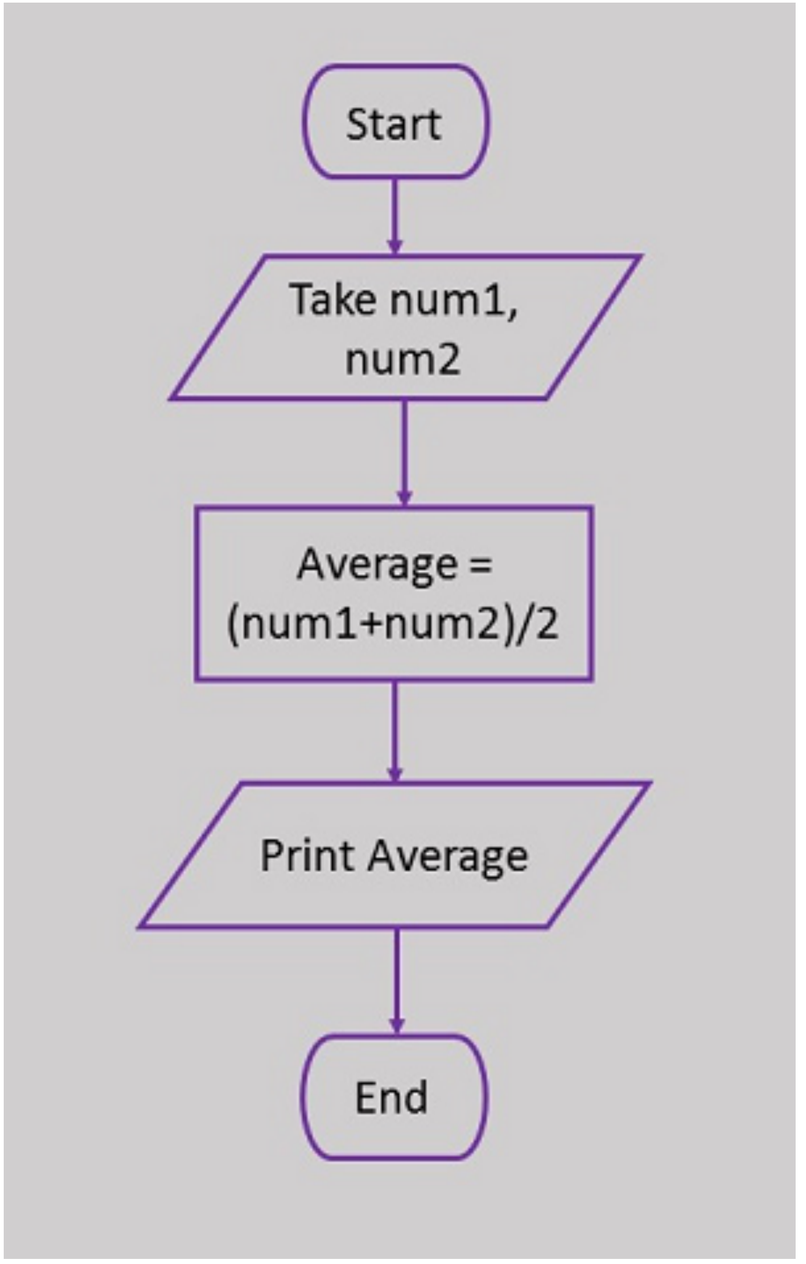
Flowchart - Example1

Draw a flowchart to represent the process of calculating the summation of two numbers



Flowchart - Example2

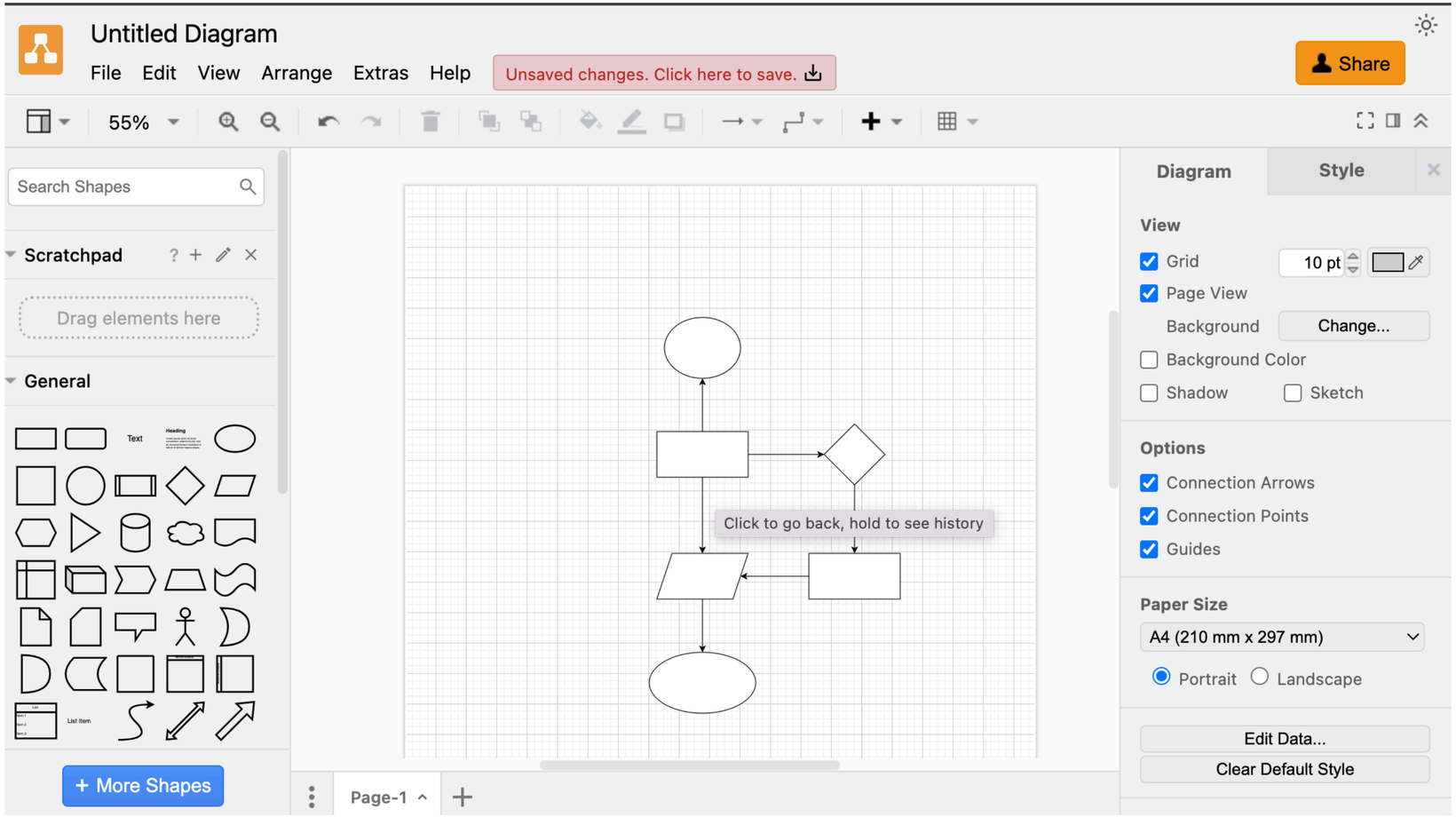
Draw a flowchart to represent the process of calculating the average of two numbers



Flowchart

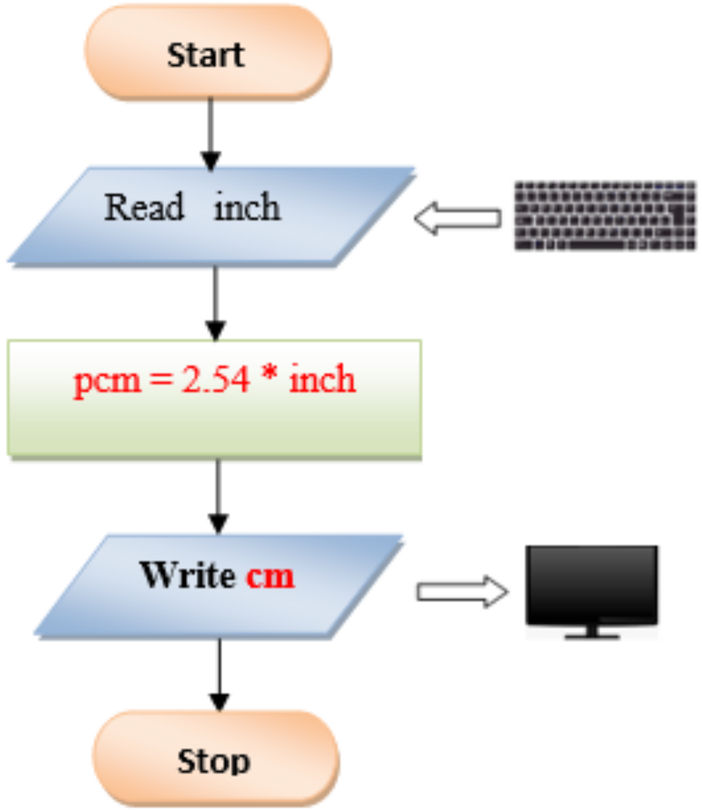
"**draw.io**" is a popular online diagramming tool that allowed users to create various types of diagrams, including flowcharts.

Through draw.io or app.diagrams.net can access to the app to create flowcharts.



Flowchart to C++

Flowchart



Program

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

int main()
{
    /*we need to give only one input
     to program i.e., inches*/

    float inch;

    float cm;
    cout<<"Enter inches:"<<endl;
    cin>>inch;

    cm = 2.54* inch;
    cout<<"Equivalent peso is:"<<cm;

    return 0;
}
```

Thank You