Al-Albayt University
Computer Science Department

C++ Programming 1 (901131)

Coordinator:

Dr. Ashraf Al-Ou'n

الفصل الدراسي الأول 2018 -2019

جامعة آل البيت
Al albayt University

Subjects

1. Introduction to C++ Programming
2. Control Structures
3. Functions
4. Arrays

1- Introduction to C++ Programming

What is computer?

- Computers are programmable devices capable of performing computations and making logical decisions.
- Computers can store, retrieve, and process data according to a list of instructions
- Hardware is the physical part of the compute: keyboard, screen, mouse, disks, memory, and processing units
- Software is a collection of computer programs, procedures and documentation that perform some tasks on a computer system

Computer Logical Units

- Input unit
  - obtains information (data) from input devices
- Output unit
  - outputs information to output device or to control other devices.
- Memory unit
  - Rapid access, low capacity, stores information
- Secondary storage unit
  - cheap, long-term, high-capacity storage, stores inactive programs
- Arithmetic and logic unit (ALU)
  - performs arithmetic calculations and logic decisions
- Central processing unit (CPU):
  - supervises and coordinates the other sections of the computer

Computer language

- Machine languages: machine dependent, it consists of strings of numbers giving machine specific instructions:

  +1300042774

  +1400593419

  +1200274027

- Assembly languages: English-like abbreviations representing elementary operations, assemblers convert assembly language to machine language:

  load basepay

  add overpay

  store grosspay

- High-level languages: Similar to everyday English, use mathematical notations, compilers convert high-level language into machine language, C++ is a high level language:

  grossPay = basePay + overTimePay

Program Design

- Programming is a creative process
- Program Design Process
  - Problem Solving Phase
    - Result is an algorithm that solves the problem
  - Implementation Phase
    - Result is the algorithm translated into a programming language

Problem Solving Phase

- Be certain the task is completely specified
  - What is the input?
  - What information is in the output?
  - How is the output organized?
- Develop the algorithm before implementation
  - Experience shows this saves time in getting program to run.
  - Test the algorithm for correctness

Problem Solving Phase

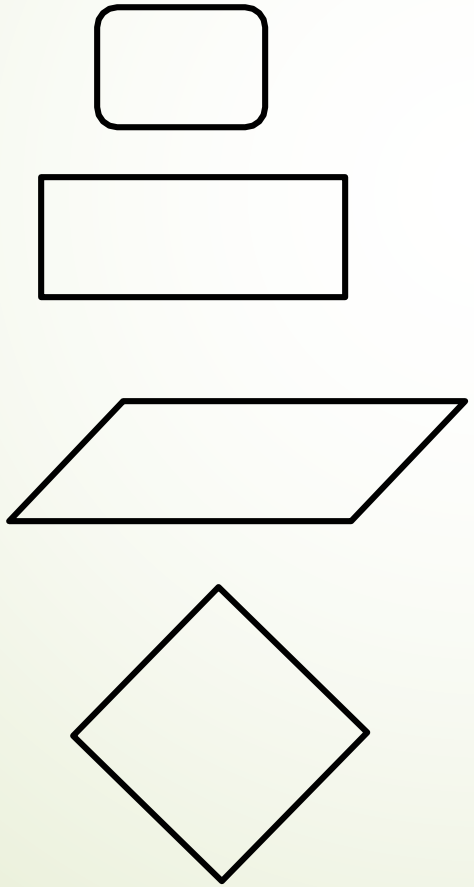
- Algorithm
  - A sequence of precise instructions (written is **pseudo** code or represented as a **flowchart**) which leads to a solution
- Pseudo code
  - Artificial, informal language similar to everyday English
  - Used to develop algorithms and not executed on computers
  - Only executable statements, no need to declare variables
- Program
  - An algorithm expressed in a language the computer can understand

Implementation Phase

- Translate the algorithm into a programming language
  - Easier as you gain experience with the language
- Compile the source code
  - Locates errors in using the programming language
- Run the program on sample data
  - Verify correctness of results
- Results may require modification of the algorithm and program

Flowchart

- Graphical representation of an algorithm or a portion of algorithm
- Drawn using certain special-purpose symbols connected by arrows called flow lines:



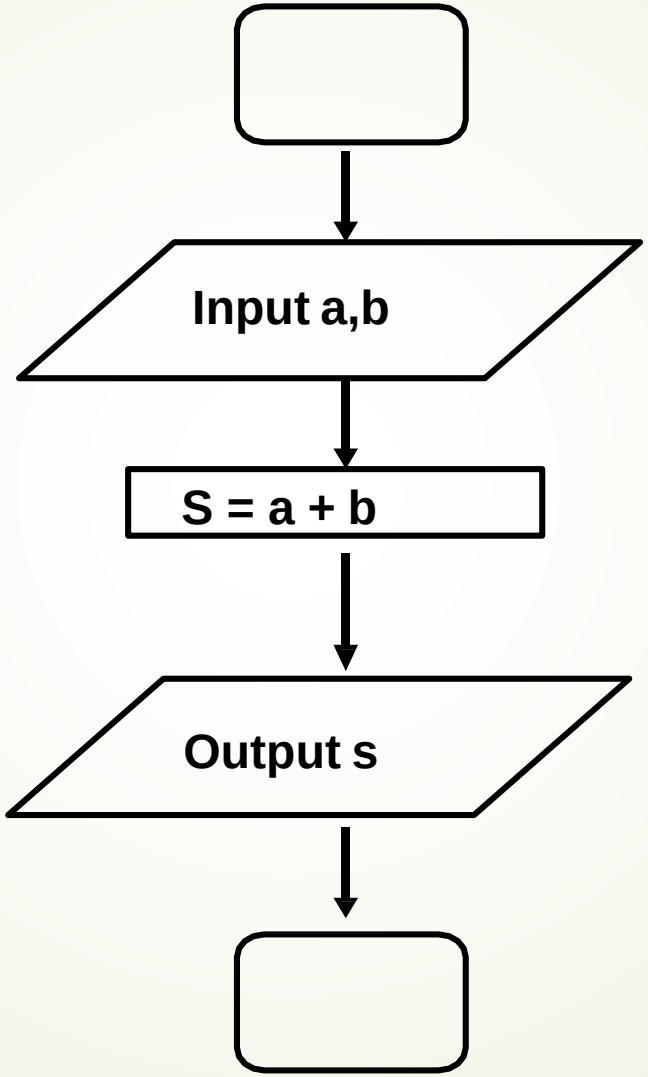
Start and End

Calculation

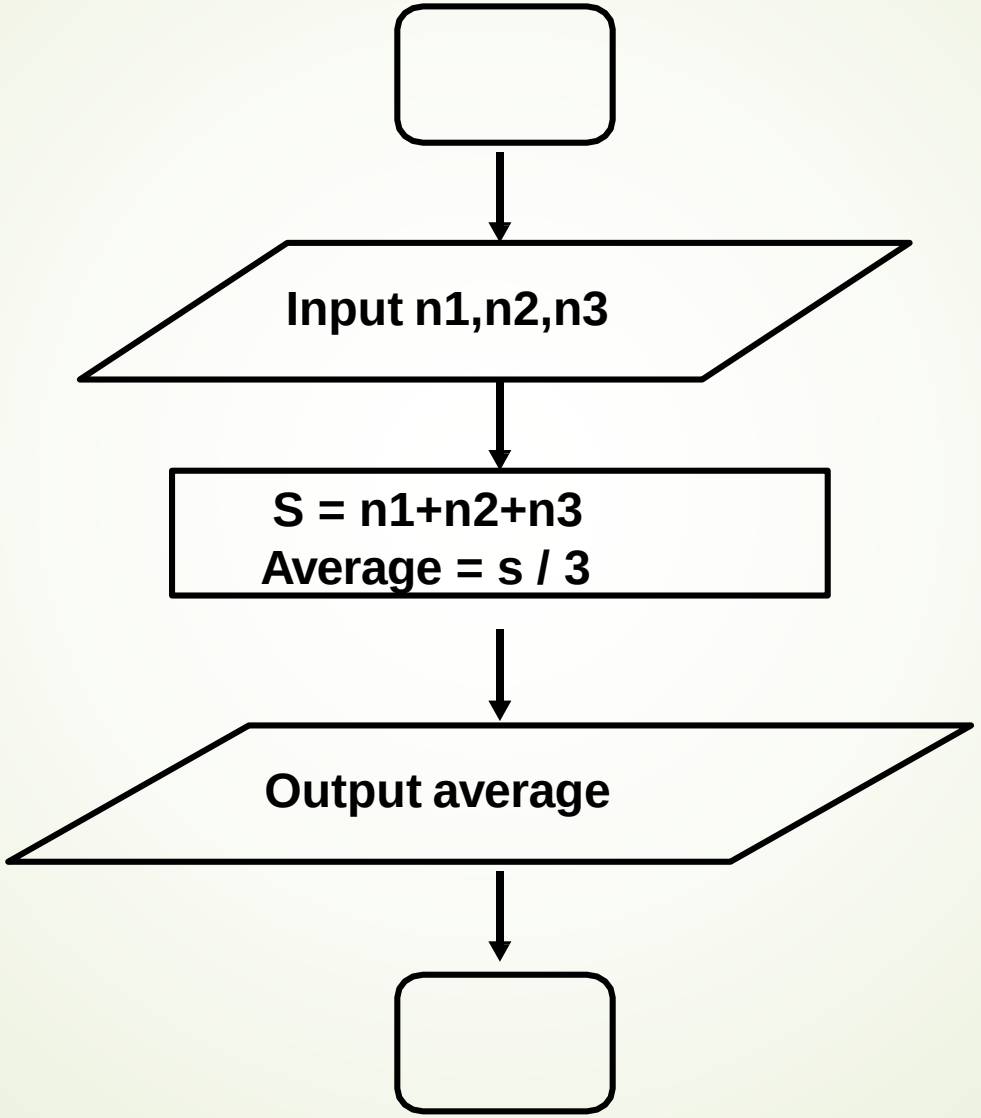
Input / output

Selection

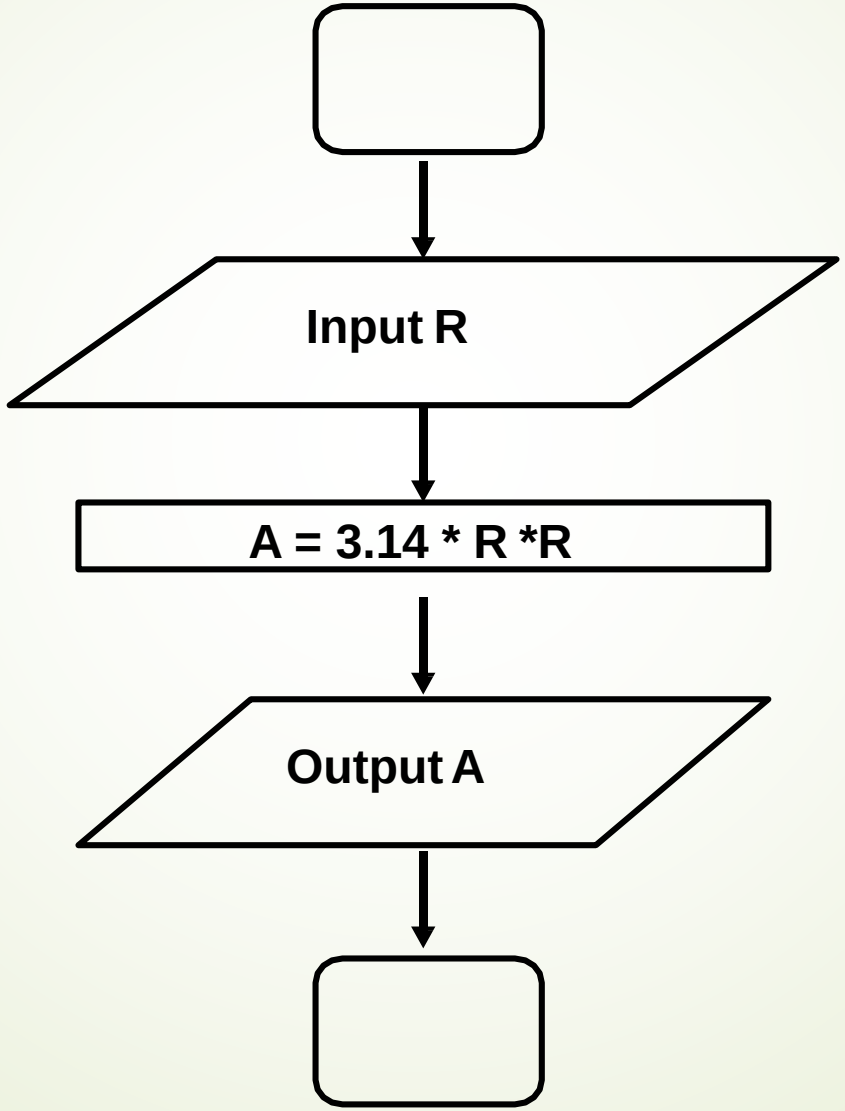
Compute and print the summation of two numbers



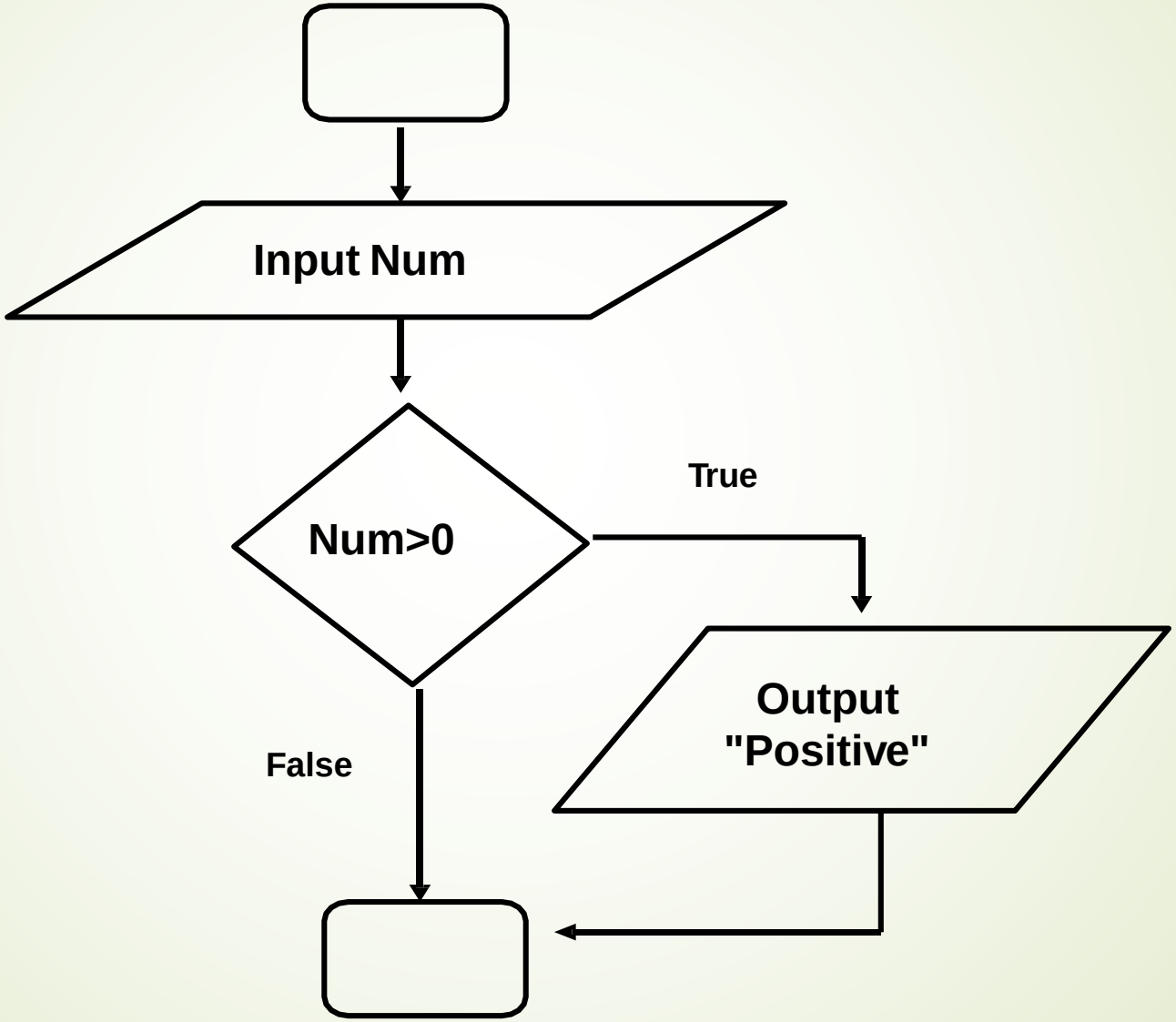
Compute and print the average of three numbers



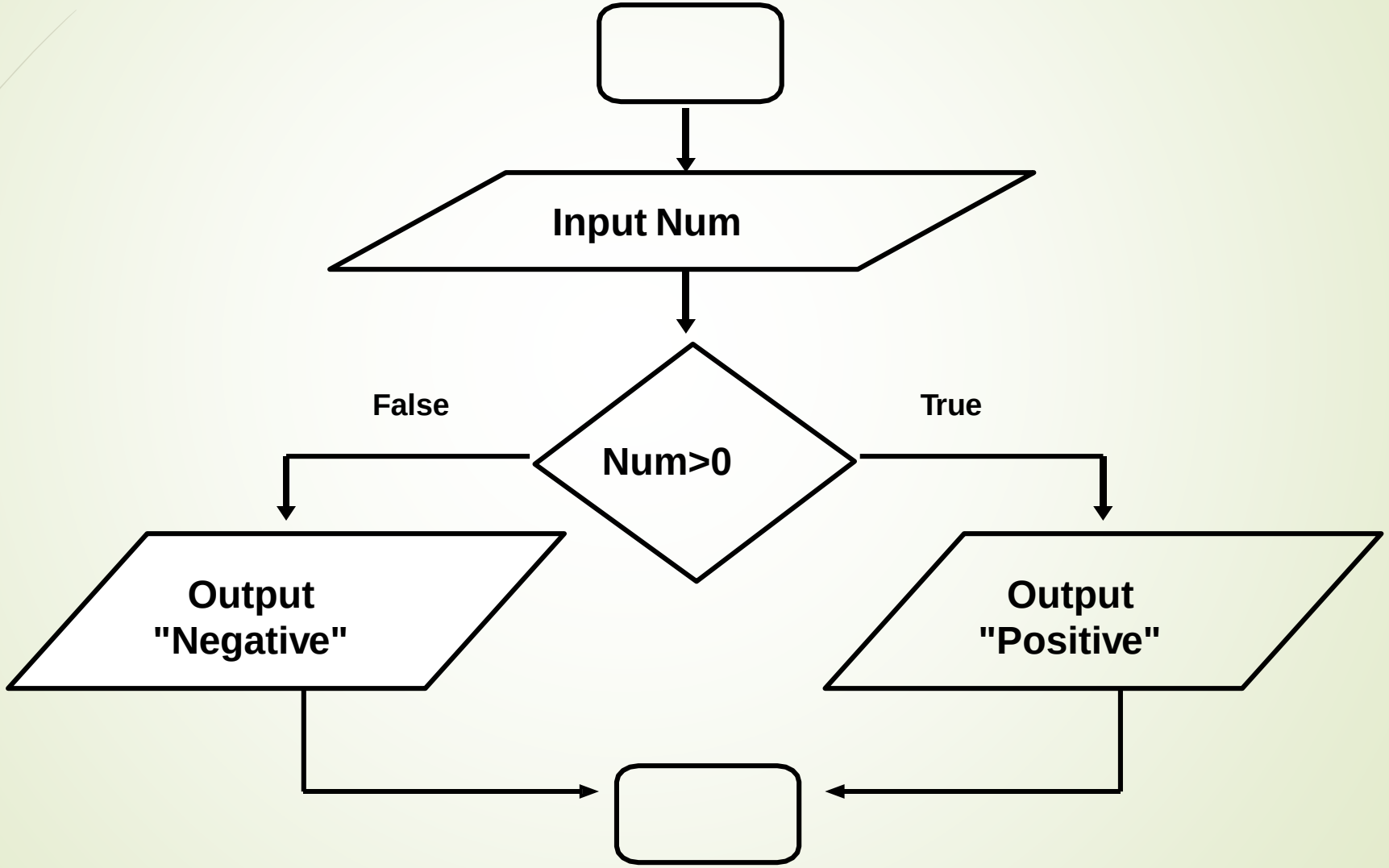
Compute the area of the circle Where area = 3.14 x R^2



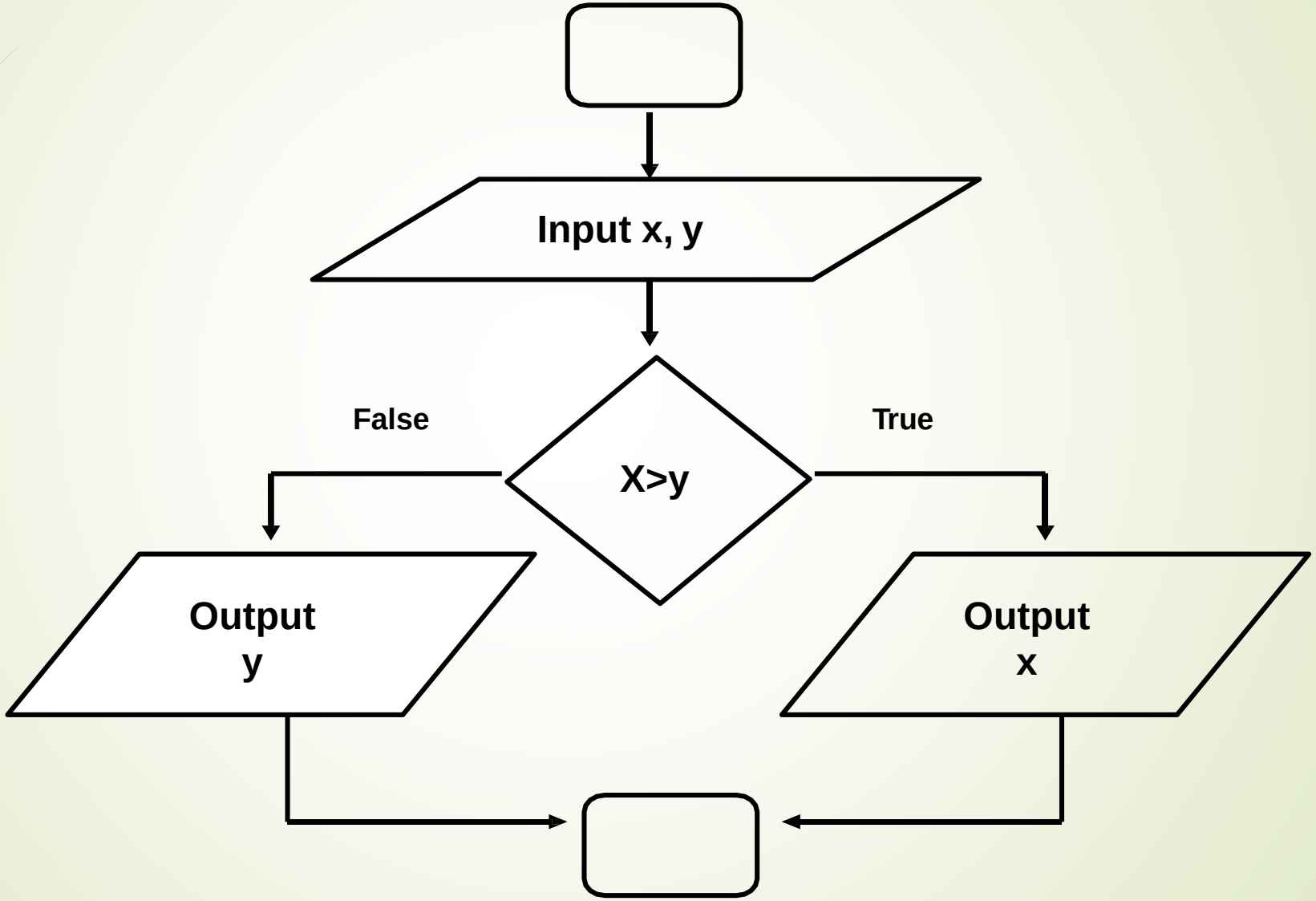
Read a number then print positive if it is positive



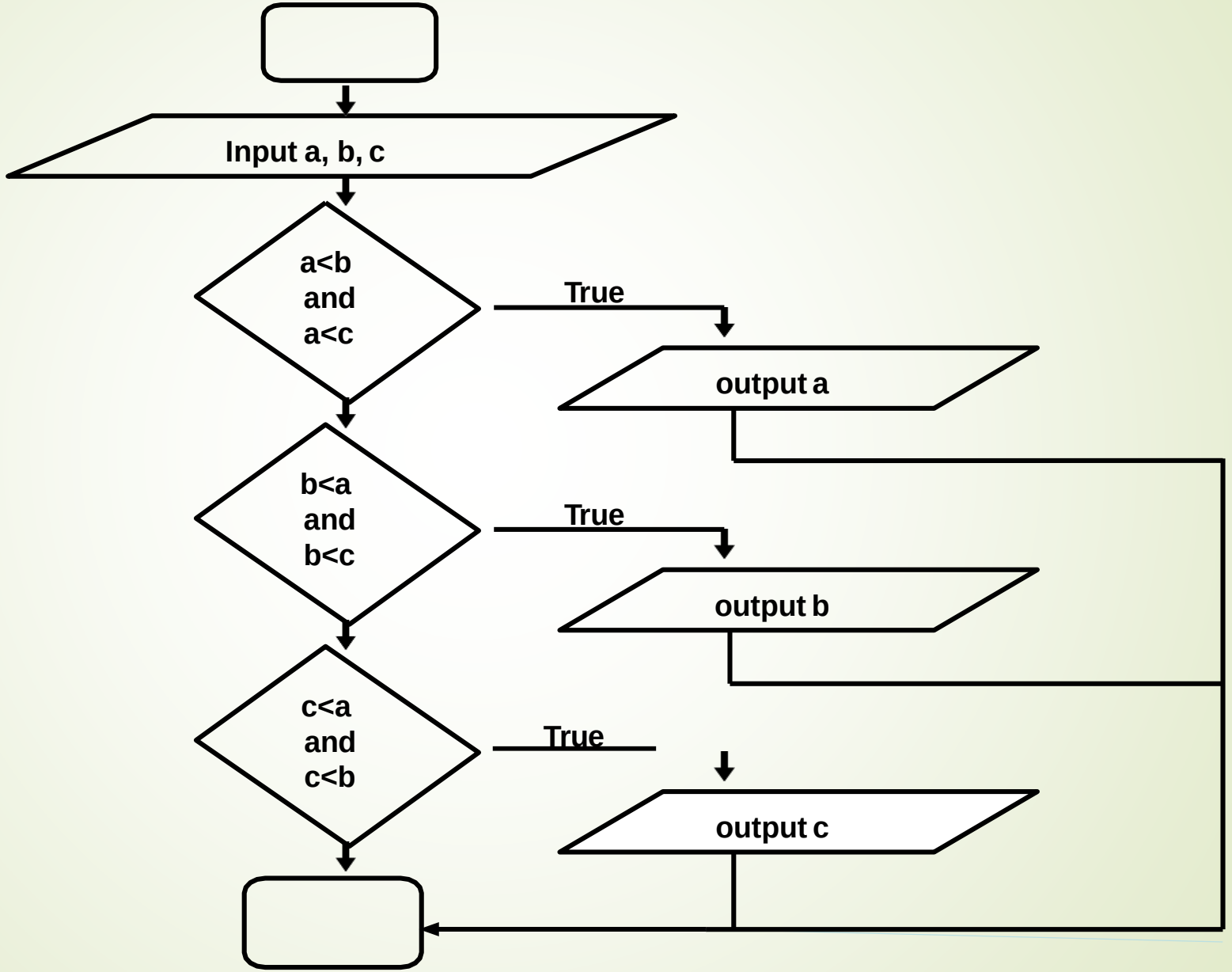
Read a number then print positive if it is positive and print negative otherwise.



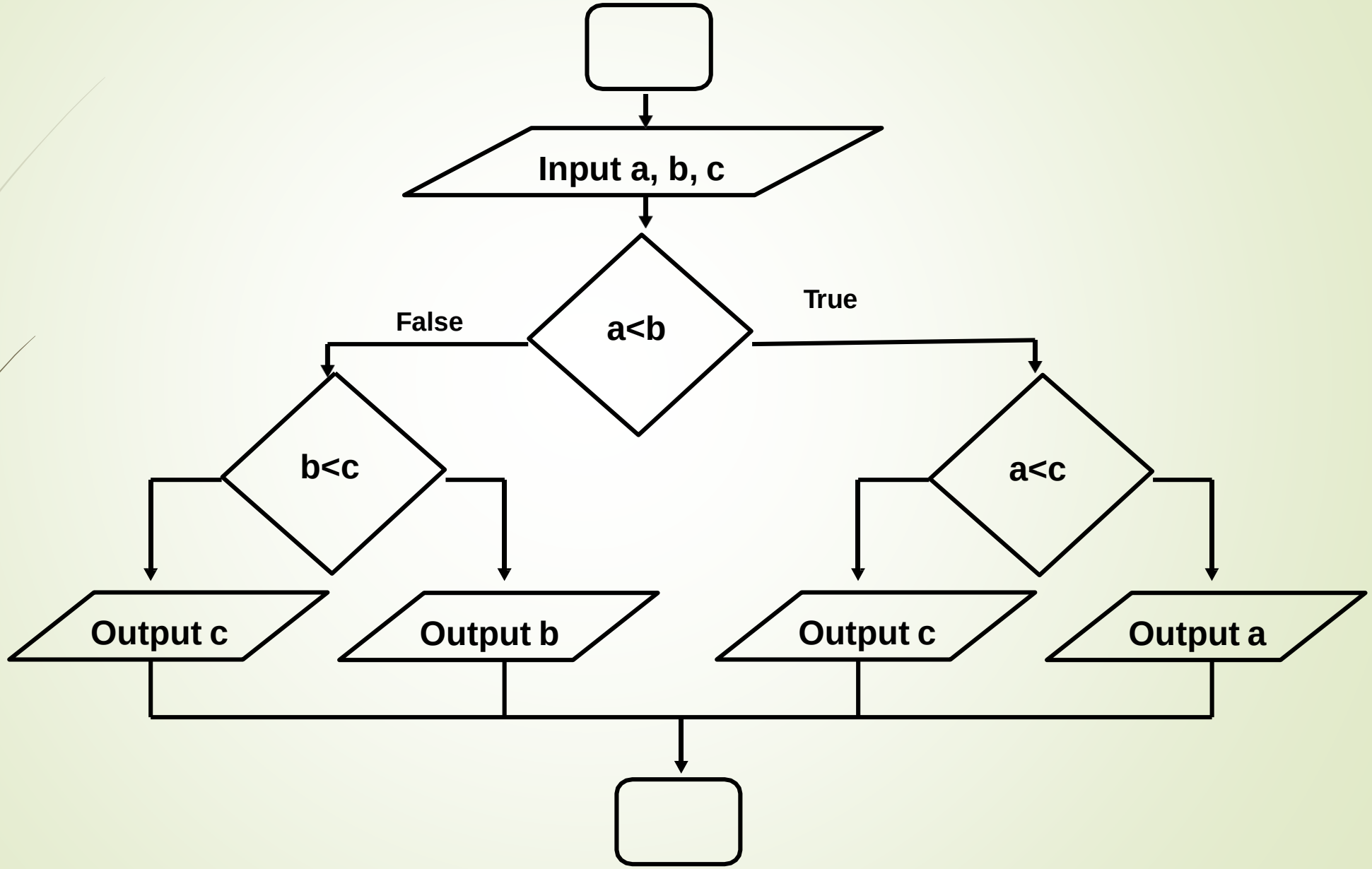
Read Two numbers then print the greatest one



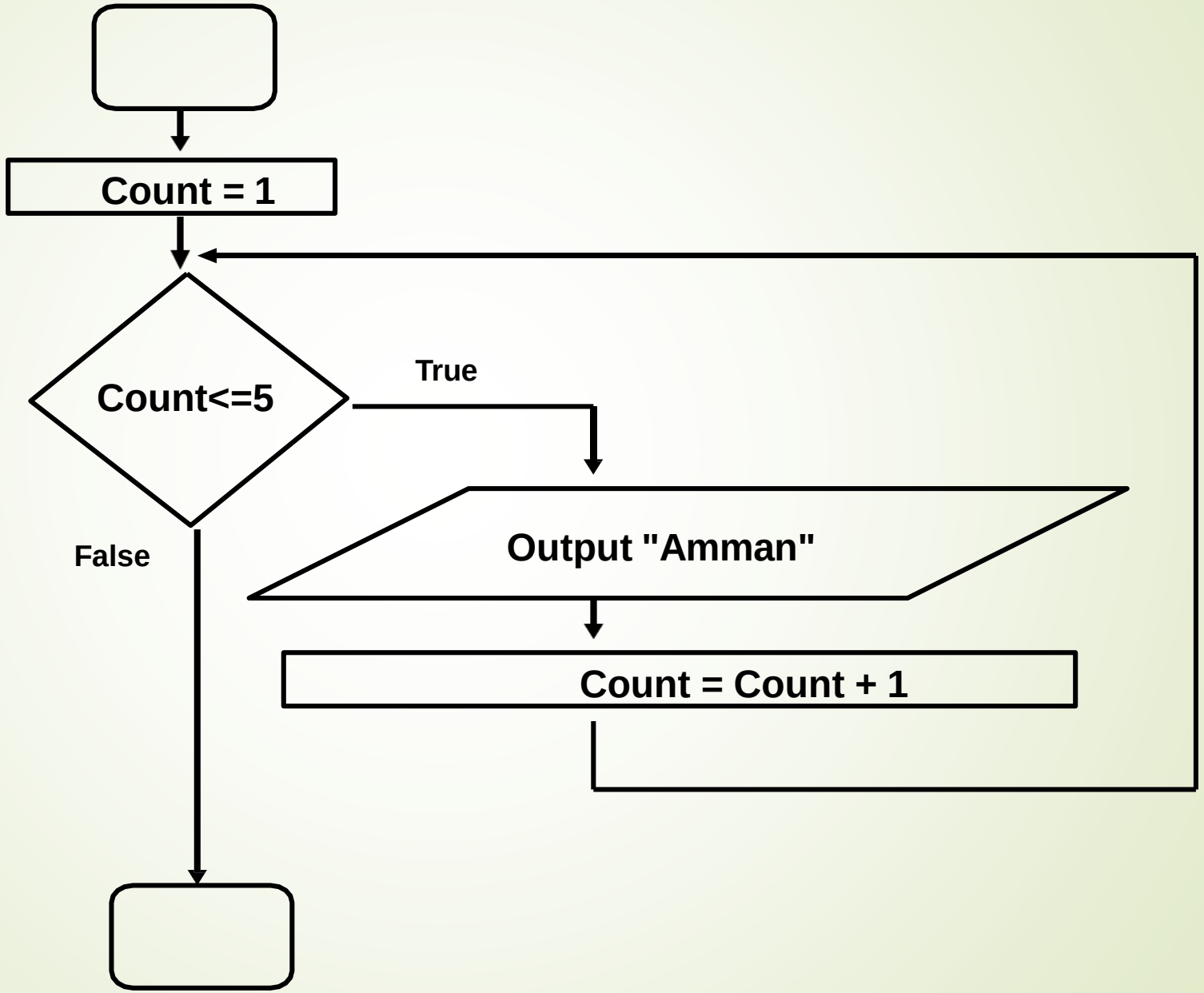
Read three numbers and print the smallest one



Another Solution

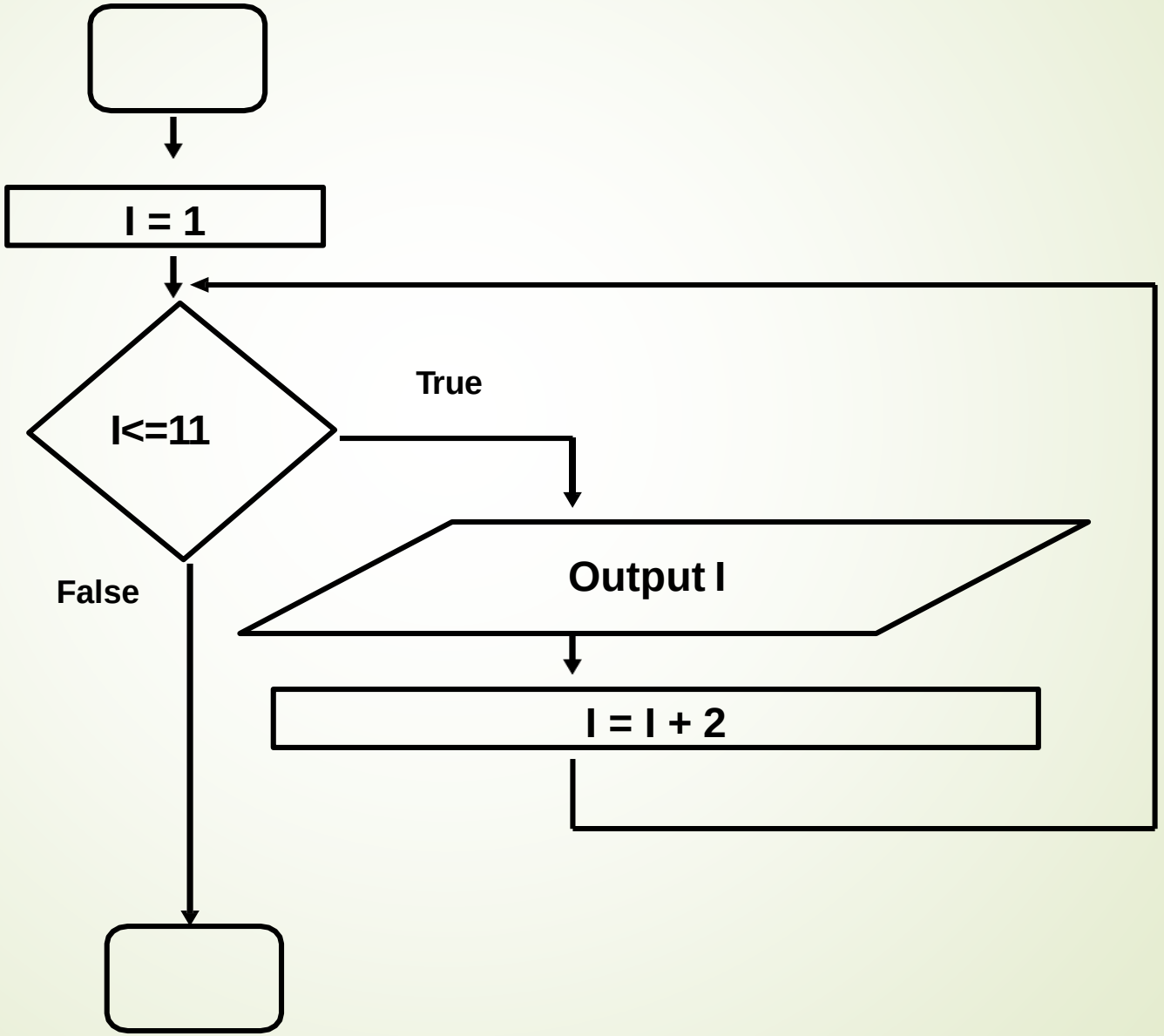


Print the word "Amman" five times.



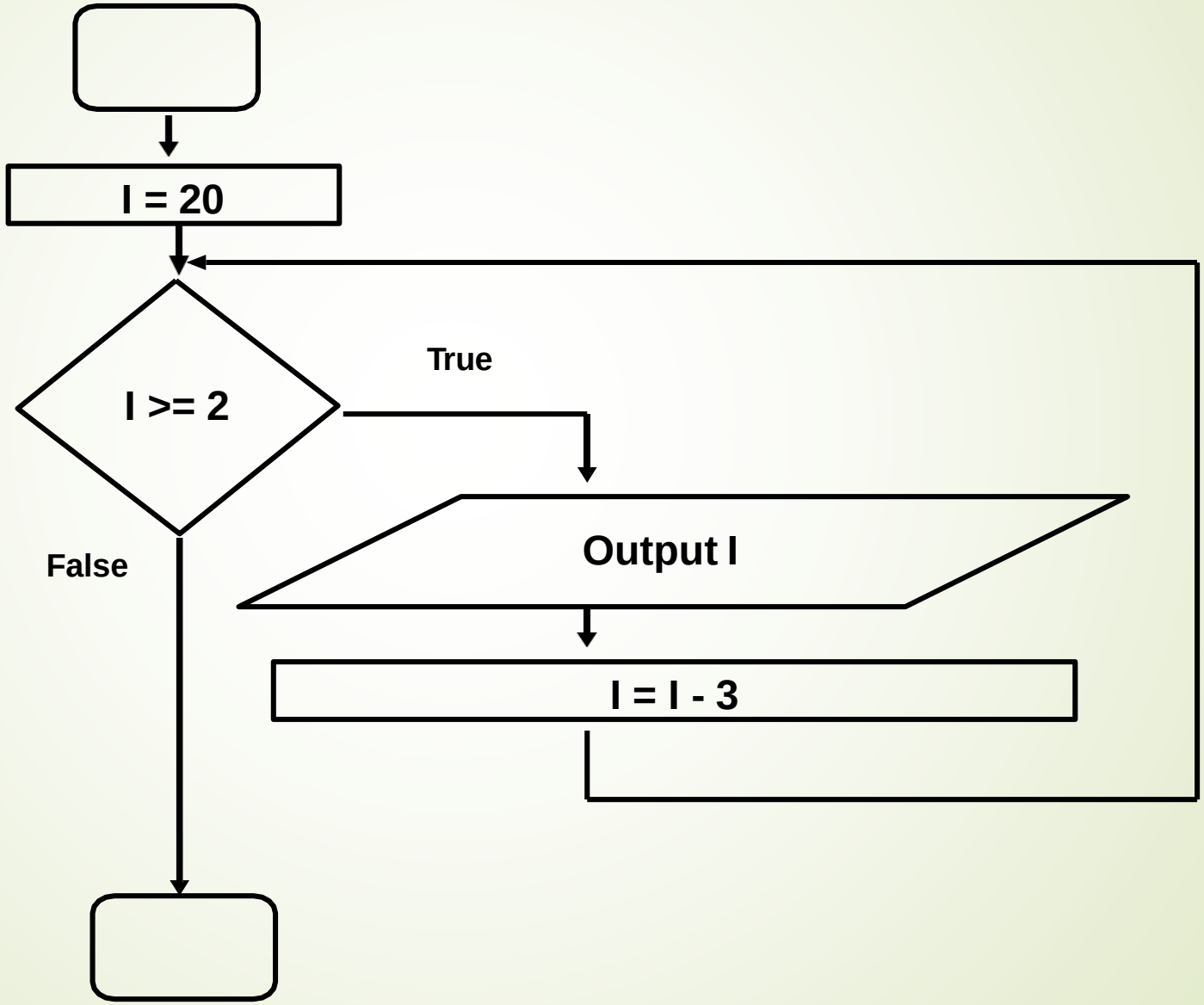
Print the following numbers

1 3 5 7 9 11

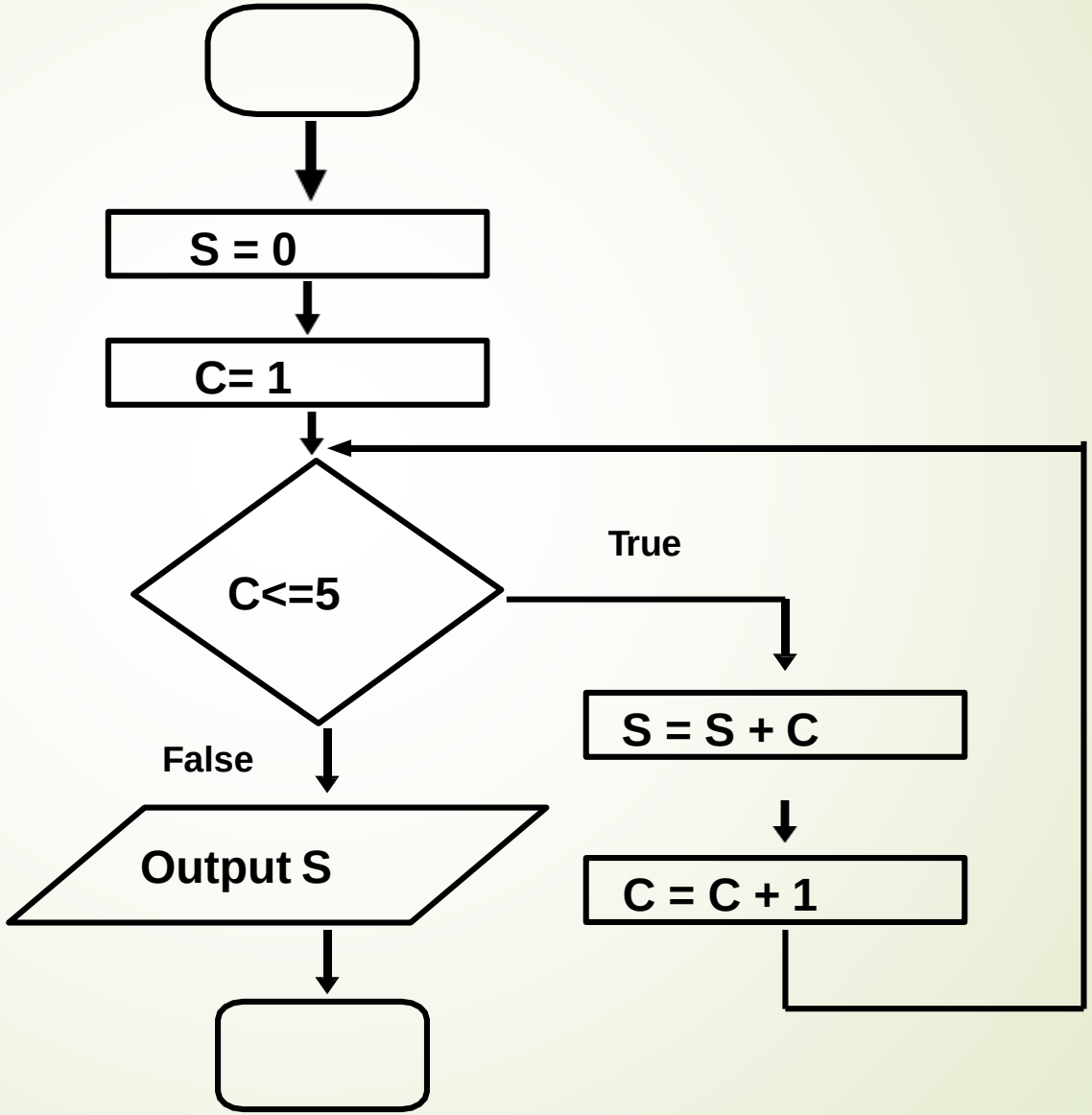


Print the following numbers

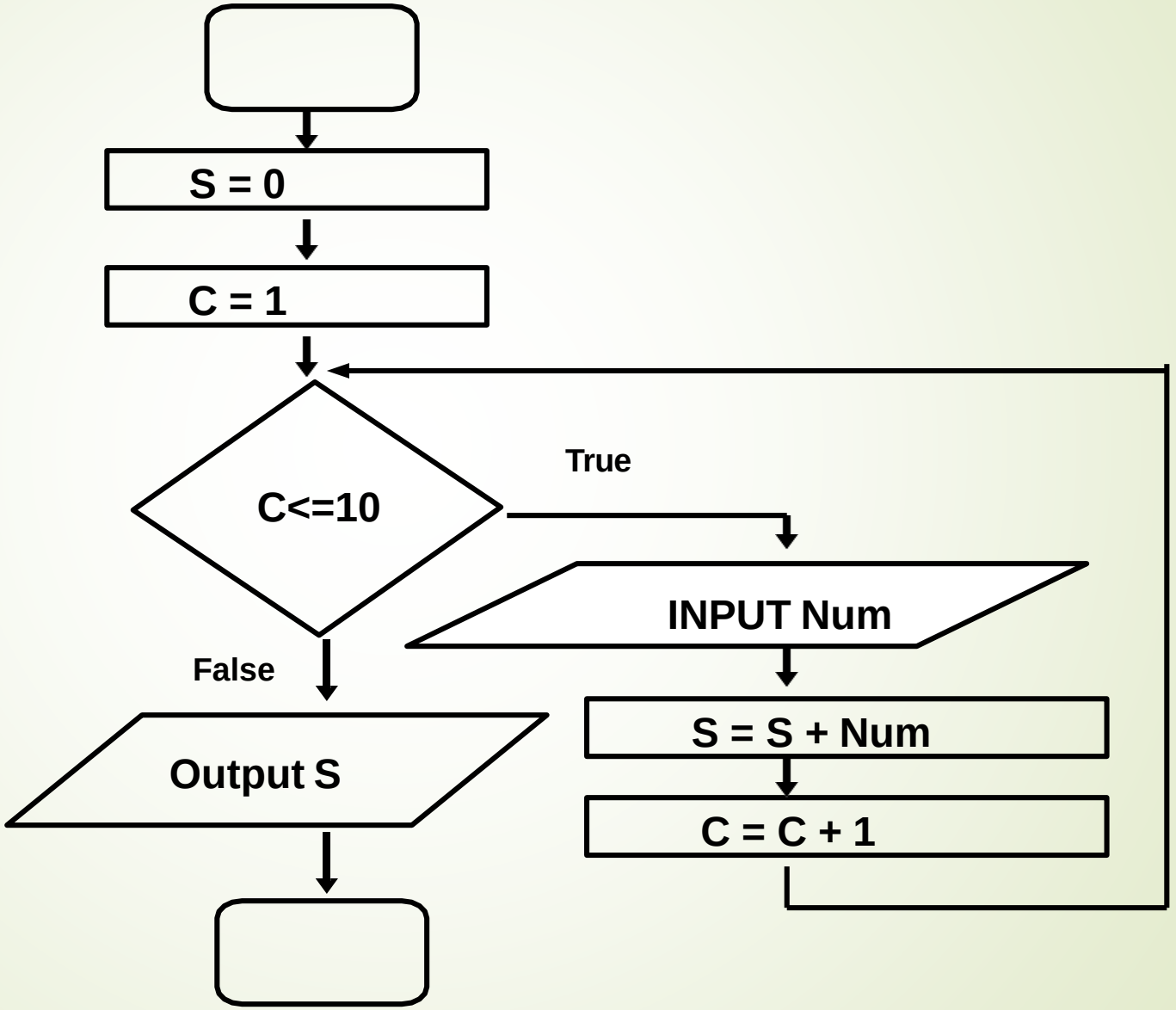
20 17 14 11 8 5 2



Compute and print S
Where $S = 1 + 2 + 3 + 4 + 5$

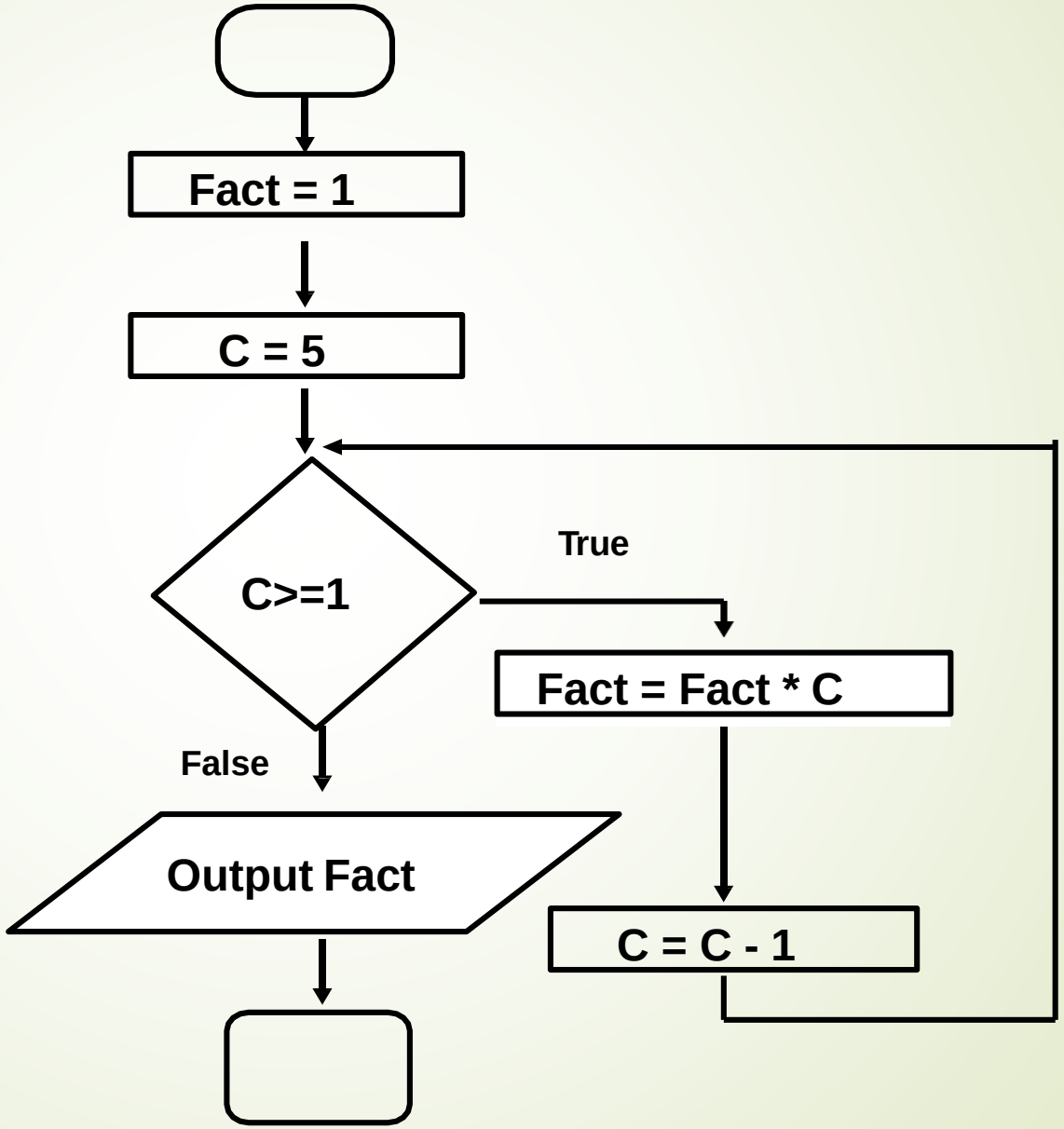


Print the Sum of 10 numbers entered by the user



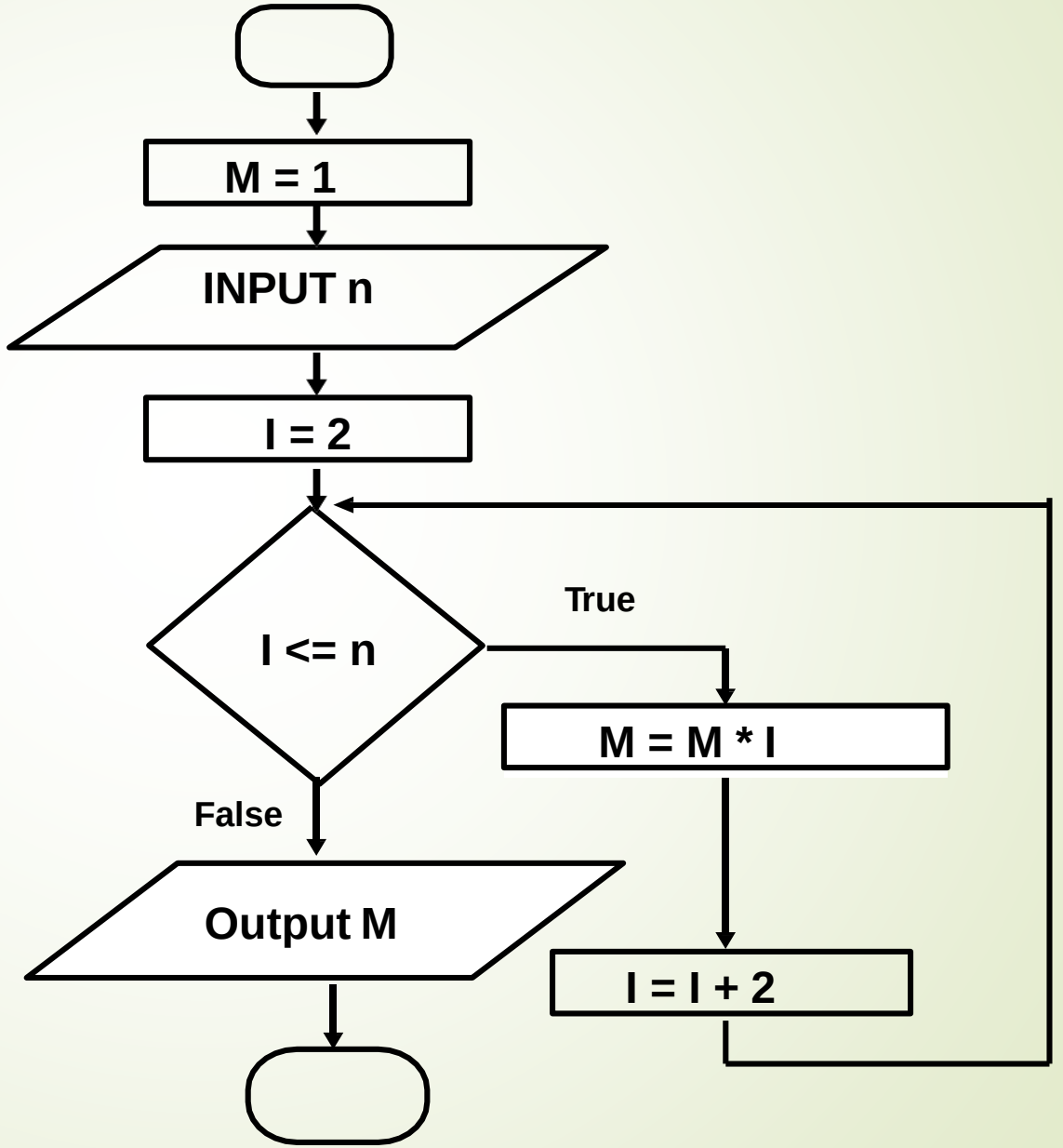
Compute and Print the factorial of 5, where:

$$fact\ (5) = 5 \times 4 \times 3 \times 2 \times 1$$



Compute and Print the value of M where:

$$M = 2 \times 4 \times 6 \times \cdots \times n$$



C++ Programming Language

- C++ standard library
  - Rich collections of existing classes and functions which are written in the core language
  - Can be used at any C++ program
- C++ programs
  - Built from pieces called **classes** and **functions which** can span multiple files
  - Structured into small understandable units to reduce the complexity and decrease program size
  - "Building block approach" to creating programs help in software reuse
- C++ is case sensitive

First C++ Program


```
// A first program in C++.
#include<iostream>

//function main begins program
execution
int main()
{
  std::cout << "Welcome to C++!\n";
}
```

Welcome to C++!

First C++ Program: Printing a Line of Text

```
// A first program in C++.
```

- Comments are ignored by compiler, used to document programs and improve readability
  - Single line comment begin with `//`, and multiple line comments begin with `/*` and end with `*/`

```
#include <iostream>
```

- Preprocessor directives begin with `#`
  - Processed by **preprocessor** before compiling
  - Causes a copy of the specified header file (`iostream`) to be included in place of the directive
  - `iosteam` is standard library header file that must be included if because `cout` is to be used

First C++ Program: Printing a Line of Text


```
int main()
```

- Part of every C++ Program
- **`main()`** is a function, which begins with left brace ({) and ends with right brace (})

```
std::cout << "Welcome to C++!\n";
```

- **`cout`** is a standard output stream object found in **`iostream`**
- **`cout`** is connected to the screen
- **`<<`** is the stream insertion operator
  - Value to right (right operand) inserted into output stream (which is connected to the screen)
- **`std::`** specifies using name that belongs to "**`namespace`**" **`std`**

- Escape characters (\): indicates "special" character output

Escape Character

Escape Sequence	Description
\n	Newline. Position the screen cursor to the beginning of the next line.
\t	Horizontal tab. Move the screen cursor to the next tab stop.
\r	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
\a	Alert. Sound the system bell.
****	Backslash. Used to print a backslash character.
\"	Double quote. Used to print a double quote character.

Example


```
#include <iostream>


int main()
{
    std::cout << "Welcome ";
    std::cout << "to C++!\n";
}
```

```
Welcome to C++!
```

Example

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

int main()
{
    cout << "Welcome\nTo\nC++!\n";
}
```

```
Welcome
to
C++!
```

Testing and Debugging

- Bug
  - A mistake in a program
- Debugging
  - Eliminating mistakes in programs

Program Errors

- Syntax errors
  - Violation of the grammar rules of the language
  - Discovered by the compiler
    - Error messages may not always show correct location of errors
- Run-time errors
  - Error conditions detected by the computer at run-time
- Logic errors
  - Errors in the program's algorithm
  - Most difficult to diagnose
  - Computer does not recognize an error

Stream extraction operator (>>)

- When used with **`cin`**, waits for the user to input a value and stores the value in the variable to the right of the operator

- The user types a value, then presses the *Enter* (Return) key to send the data to the computer

- Example:

```
int  myVariable;
cin >> myVariable;
```

  - Waits for user input, then stores input in **`myVariabl`**

Compute and print the summation of two numbers


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

int main(){
    int num1, num2, sum;
    cout <<"Please Enter two numbers:\n";
    cin >> num1 >> num2;
    sum = num1 + num2;
    cout<<"sum = "<<sum<<endl;
}
```

```
Please Enter two numbers:
2
3
sum = 5
```

Fundamental C++ Objects

- Integer objects

  `short` `int` `long`

- Floating-point objects

  `float` `double` `long double`

  – represent real numbers

- Character objects

  `char`

  – may hold only a single letter, a single digit, or a single special character like a, $, 7, *.

- Different types allow programmers to use resources more efficiently

Character object type

- ASCII is the dominant encoding scheme
  - `' '` encoded as 32 `'+'` encoded as 43
  - `'A'` encoded as 65 `'a'` encoded as 97
- Explicit (literal) characters within single quotes:

```
'a'  'D'  '*'
```

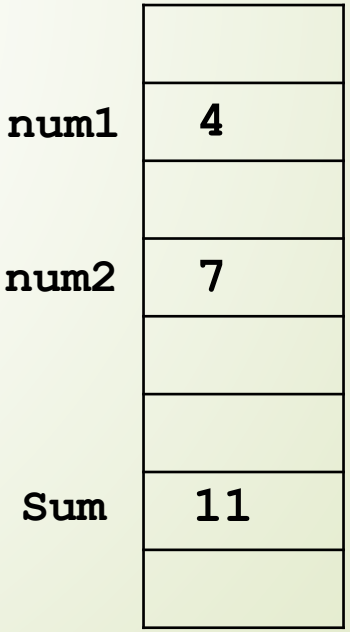
- Special characters - delineated by a backslash \

```
'\n'  '\t'  '\\'
```

Memory Concepts

- Variables are names of memory locations
- Correspond to actual locations in computer's memory
- Every variable has name, type, size and value
- When new value placed into variable, overwrites previous value
- Reading variables from memory is nondestructive

```
int num1 = 4;

int num2 = 7;

int sum = num1 + num2;
```



Names (Naming Entities)

- Used to denote program values or components
- A valid name is a sequence of
  - Letters (upper and lowercase)
  - A name cannot start with a digit
- Names are case sensitive
  - MyVar is a different name than MYVAR
- There are two kinds of names
  - Keywords
  - Identifiers

Keywords

- Keywords are words reserved as part of the language
- They cannot be used by the programmer to name things
- They consist of lowercase letters only
- They have special meaning to the compiler

C++ Keywords

and	continue	goto	public	try
and_eq	default	if	register	typedef
asm	delete	inline	reinterpret_cast	typeid
auto	do	int	return	typename
bitand	double	long	short	union
bitor	dynamic_cast	mutable	signed	unsigned
bool	else	namespace	sizeof	using
break	enum	new	static	virtual
case	explicit	not	static_cast	void
catch	export	not_eq	struct	volatile
char	extern	operator	switch	wchar_t
class	false	or	template	while
compl	float	or_eq	this	xor
const	for	private	throw	xor_eq
const_cast	friend	protected	true	

Identifiers

- Used to name entities in C++
- Consists of letters, digits or underscore
  - Starts with a letter or underscore
  - Can not start with a digit
- Identifiers should be:
  - Short enough to be reasonable to type
  - Long enough to be understandable
- Examples
  - **`Grade`**
  - **`Temperature`**
  - **`CameraAngle`**
  - **`IntegerValue`**

Definitions/declaration

- All variable that are used in a program must be defined (declared)
- A variable definition specifies Type and Identifier
- General definition form: **`Type Id;`**
- Examples:

```
Char Response;
int MinElement;
float Score;
float Temperature;
int  i;
char c;
double x;
```

- Value of a variable is whatever in its assigned memory location
- Memory location is where a variable value can be stored for program use

Type compatibilities

- Store the values in variables of the same type
- This is a type mismatch:

```
int x;
x = 2.99;
```

- Variable x will contain the value 2, not 2.99

Arithmetic

- Arithmetic is performed with operators.
- Arithmetic operators are listed in following table

C++ operation	Arithmetic operator	Algebraic expression	C++ expression
Addition	**+**	***f + 7***	**f + 7**
Subtraction	**-**	***p – c***	**p – c**
Multiplication	*****	***bm***	**b * m**
Division	**/**	***x / y***	**x / y**
Modulus	**%**	***r mod s***	**r % s**

- Modulus operator returns the remainder of integer division

  `7 % 5` evaluates to 2

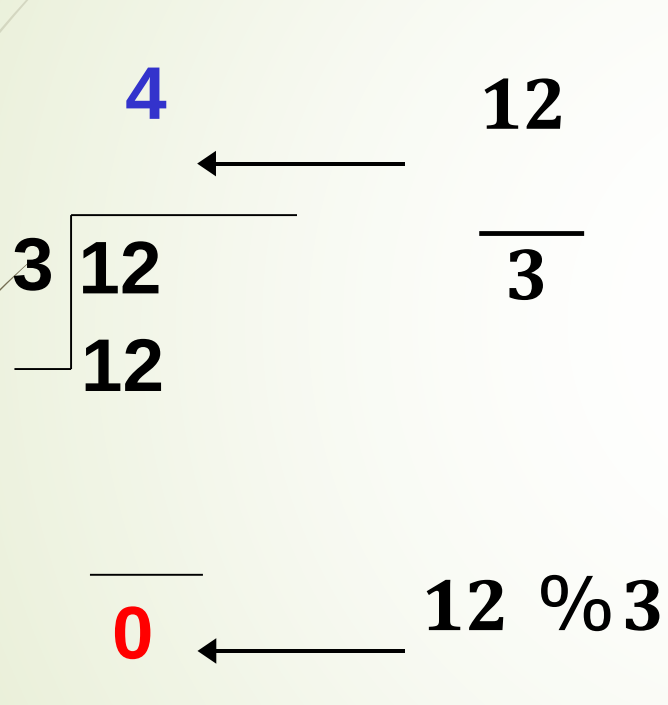
- Integer division truncates remainder

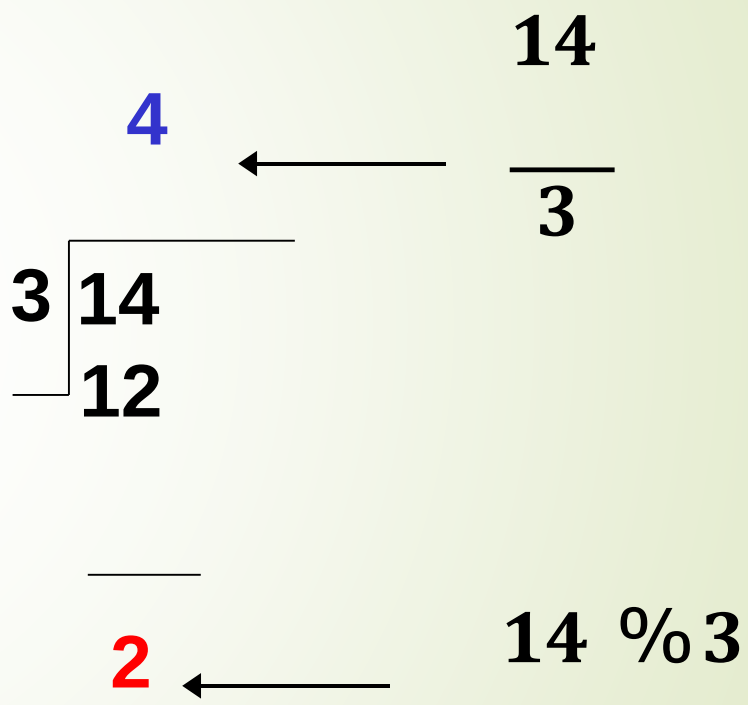
  `7 / 5` evaluates to 1

Results of Arithmetic operators

- Arithmetic operators can be used with any numeric type.
- An **operand** is a number or variable used by the operator e.g.
  - integer1 + integer2
    - **+** is operator
    - **integer1** and **integer2** are operands
- Result of an operator depends on the types of operands
  - If both operands are `int`, the result is `int`
  - If one or both operands are `double`, the result is `double`

Integer Division





Examples on integer division


```
#include <iostream>
using namespace std;
int main( )
{
  cout<< 10/4 <<endl;
  cout<< 10.0/4 <<endl;
  cout<< 10/4.0 <<endl;
}
```

```
2
2.5
2.5
```

Comparing mathematical and C++ expressions

Mathematical formula	C++ Expression
$x^2 - 5yz$	$x * x - 5 * y * z$
$x(y + 2z)$	$x * (y + 2 * z)$
$\frac{1}{x^2 + 4y + 3}$	$1/(x * x + 4 * y + 3)$
$\frac{w + x}{y + 2z}$	$(w + x)/(y + 2 * z)$

Operator precedence

- The order in which an operator is executed
- For example, the multiplication operator (*) is executed before addition operator (+)
- To find the average of three variables a, b and c
  - **`Incorrect: a + b + c / 3`**
  - **`Correct: (a + b + c ) / 3`**

Rules of operator precedence

Operator(s)	Operation(s)	Order of evaluation (precedence)
`()`	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
`*, /, or %`	Multiplication Division Modulus	Evaluated second. If there are several, they are evaluated left to right.
`+ or -`	Addition Subtraction	Evaluated last. If there are several, they are evaluated left to right.

Example on Operator Precedence

Evaluate the following arithmetic expression:

```
20 - 10 / 5 * 2 + 3 * 5 % 4

1)10/5     =   2  →   20 - 2   *  2   +  3  *  5  %  4

2)2*2      =  4    →   20 - 4   +  3   *  5  %  4

3)3*5      =  15   →   20 - 4   +  15   %  4

4)15%4     =   3  →   20 - 4   +  3

5)20    - 4    =  16    → 16   + 3
6)16    + 3    =  19
```

Assignment operator (=)

- The (=) operator in C++ is not an equal sign. It assigns a value to a variable
- An assignment statement changes the value of the variable on the left of the assignment operator (=)
- General Form: ***identifier= expression;***
- On the right of the assignment operator can be
  - Constant: **`x = 21;`**
  - Variable: **`x = y;`**
  - Expression: **`x = y * 2 + z;`**

- The following statement is not true in algebra: **`i = i + 3;`**
  - In C++ it means the **new** value of **`i`** is the **previous** value of **`i`** plus 3

Assignment expression abbreviations

- C++ provides several assignment operators for abbreviating assignment expressions, as shown in the table below:

Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
`+=`	`c += 7`	`c = c + 7`	`10` to `c`
`-=`	`d -= 4`	`d = d - 4`	`1` to `d`
`*=`	`e *= 5`	`e = e * 5`	`20` to `e`
`/=`	`f /= 3`	`f = f / 3`	`2` to `f`
`%=`	`g %= 9`	`g = g % 9`	`3` to `g`

Print the average of three numbers


```
int main() {
      int n1 , n2 , n3;
      float s , average;
      cout << "Please Enter three integers:\n";
      cin >> n1 >> n2 >> n3;
      s = n1 + n2 + n3;
      average = s / 3;
      cout << "Average = " << average << endl;
}
```

```
Please Enter three integers:
1
6
2
Average = 3
```

Compute the area of a circle, where area = π x r^2


```
int main(){

    double Pi = 3.14;
    int r;

    cout<<"Please enter r : ";
    cin>>r;

    double area;
    area = Pi * r * r;

    cout<<"Circle's Area = "<< area <<endl;
}
```

Increment and Decrement Operators

- Increment and decrement operators are unary operators as they require only one operand.
  - ++ unary increment operator: Adds 1 to the value of a variable
  - -- unary decrement operator
  - **`x++`** is equivalent to **`x = x + 1`**
  - **`x--`** is equivalent to **`x = x – 1`**

- Pre-increment
  - When the operator is used before the variable (**++c**), Variable is changed, then the expression it is in is evaluated

- Post-increment
  - When the operator is used after the variable (**c++),** Expression the variable is in executes, then the variable is changed.

Increment and Decrement Operators

- Example: If `c = 5`, then
  - `cout << ++c;`
    - **c** is changed to **6**, then printed out
  - `cout << c++;`
    - Prints out **5** (**cout** is executed before the increment)
    - **c** then becomes **6**
- When variable not in expression
  - Preincrement and postincrement have same effect

```
++c;
cout << c;
```

and

```
c++;
cout << c;
```

are the same

Increment and decrement operators

Operator	Sample expression	Explanation
++ Preincrement	**++a**	Increment **a** by 1, then use the new value of **a** in the expression in which **a** resides.
++ Postincrement	**a++**	Use the current value of **a** in the expression in which **a** resides, then increment **a** by 1.
-- Predecrement	**--b**	Decrement **b** by 1, then use the new value of **b** in the expression in which **b** resides.
-- postdecrement	**b--**	Use the current value of **b** in the expression in which **b** resides, then decrement **b** by 1.

Understand the effect of pre and post-increment


```
int main(){

   int c;
   c = 5;
   cout << c << endl;
   cout << c++ << endl
   cout << c << endl << endl;

   c = 5;
   cout << c << endl;
   cout << ++c << endl;
   cout << c << endl;
}
```

```
5
5
6


5
6
6
```

Operators Precedence

Operators	Associativity	Type
`()`	left to right	parentheses
`++ --`	left to right	unary (postfix)
`++ -- + -`	right to left	unary (prefix)
`* / %`	left to right	multiplicative
`+ -`	left to right	additive
`< <= > >=`	left to right	relational
`== !=`	left to right	equality
`= += -= *= /= %=`	right to left	assignment

Example


```
int main(){
    int X = 5 , Y = 7 , Z;

    cout<< X++ <<endl;
    cout<< ++X <<endl;

    Z = X++ + Y;
    cout <<X<<"\t"<<Y<<"\t"<<Z<<endl;

    Z = ++X + Y;
    cout <<X<<"\t"<<Y<<"\t"<<Z<<endl;

    Z = X++ + Y++;
    cout <<X<<"\t"<<Y<<"\t"<<Z<<endl;
}
```

```
5
7

8	7	14

9	7	16

10	8	16
```

2 - Control Structures

Control Structures

- Sequence structure: C++ programs executed sequentially by default.
- Selection structures
  - `if` selection structure
    - Perform an action if condition is true.
    - Skip the action if condition is false.
  - `if/else` selection structure
    - Perform an action if condition is true.
    - Performs a different action if the condition is false.
  - `switch` selection structure
    - Perform one of many different actions depending on the value of an integer expression.

Control structures

- Repetition structures
  - **`while`** repetition structure
    - An action to be repeated while some conditions remains true.
  - **`do/while`** repetition structure
    - Similar to while structure.
    - Tests the loop continuation after the loop body is performed.
    - **`while`** tests the loop continuation condition before the loop body is performed.
  - **`for`** repetition structure
    - used when the number of times to be repeated is fixed/known
    - It handles all the details of the counter controlled repetition
    - Execution continues as long as the condition is true

Condition

- **Condition** is a logical expression that evaluates to **`true`** or **`false`**
- Specifies the decision you are making
- Conditions can be formed by using the equality **(==)** and relational operators **(< , > , >= , <= , !=)**
- Equality operators precedence is lower then precedence of relational operators.

Arithmetic Expressions

- Composed of operands and arithmetic operations (+ , - , *, /, %)
- evaluates to a numeric value
  - (e.g. 3 + 4 gives 7)
- Operands may be numbers and/or identifiers that have numeric values

Relational Expressions

- Composed from operands and operators.
- Operands may be numbers and/or identifiers that have numeric values.
- Result is a logical value (**`true`** or **`false`**).
- Operators are relational operators: < , > , <= , >= ,= =, !=
- Example:
  - **`(a < b)`** gives **`true`** if value of **`a`** is less than value of **`b`**, or gives **`false`** if value of **`a`** is not less than value of **`b`**
  - **`(x != y)`** gives **`true`** if **`x`** does not equal **`y`** or gives **`false`** if **`x`** equal **`y`**

Equality and relational operators

Standard algebraic equality operator or relational operator	C++ equality or relational operator	Example of C++ condition	Meaning of C++ condition
Relational operators			
>	>	`x > y`	**x is greater than y**
<	<	`x < y`	**x is less than y**
≥	>=	`x >= y`	**x is greater than or equal to y**
≤	<=	`x <= y`	**x is less than or equal to y**
Equality operators			
=	==	`x == y`	**x is equal to y**
≠	!=	`x != y`	**x is not equal to y**

Boolean variables andrelational operations


```
int main( )  {
      bool x , y;
      x = 5 > 7;
      cout << "x = " << x << endl;
      y =  5 < 7;
      cout << "y = " << y << endl;
      x = true;
      cout << "x = " << x << endl;
      y = false;
      cout << "y = " << y << endl;
      x = 5;
      cout << "x = " << x;
}
```

```
x = 0

y = 1

x = 1

y = 0

x = 1
```

Logical Expressions

- Also called **Boolean expressions**
- Result is a logical value **`true`** or **`false`**
- Composed from operands and operators.
- Operands are identifiers that have logical values
- Operators are logical operators:
  - **`&&`(`AND`)**
  - **`||`(`OR`)**
  - **`!`(`NOT`)**
- Example:
  - **`X && Y`**
  - **`a && b || c`**

Evaluating Logical Expressions

- AND truth table

&&		
True	True	True
True	False	False
False	True	False
False	False	False

- OR truth table

||		
True	True	True
True	False	True
False	True	True
False	False	False

- NOT truth table

!	
True	False
False	True

Arithmetic, Relational and Logical Expressions

- Relational expression may contain arithmetic sub expressions:
  - `( 3 + 7 ) <(12 * 4 )`

- Logical expression may contain relational and arithmetic subexpressions:
  - `x && y && (a > b)`
  - `(2 + t) < (6 * w) && (p == q)`

Operators Precedence

Operators	Associativity	Type
`()`	left to right	parentheses
`++ --`	left to right	unary (postfix)
`++ -- + -`	right to left	unary (prefix)
`* / %`	left to right	multiplicative
`+ -`	left to right	additive
`< <= > >=`	left to right	relational
`== !=`	left to right	equality
`&&`		
`||`		
`= += -= *= /= %=`	right to left	assignment

Operators Precedence


```
int main( )
{
   int a = 10, b = 3;

  cout<<"a+b="<<a+b <<endl;
  cout<<"a+b*2= "<<a+b*2 <<endl;

  cout<<"(a+b)*2= "<<(a+b)*2<<endl;

  cout<<a<<"<"<<b<<"is " << (a<b)
      <<endl;

  cout<<"a+b != a+3 is "
      <<(a+b != a+3);

}
```

```
a+b=13

a+b*2= 16

(a+b)*2= 26

10<3 is 0

a+b != a+3 is 0
```

`if` selection Structure

```
if ( Condition )
     statement;


if ( Condition )
{
     statement1;
     statement1;
     statement1;
     …
}
```

Read any number from user, then print positive if it is positive

```
int main()
{
    int Num;

    cout<<"Enter an integer Number:";

    cin >> Num;

    if (Num > 0)
        cout<<" Positive\n";
}
```

Another Version


```
int main(){

  int Num;
  bool w;

  cout<<"Enter an integer number:";
  cin >> Num;

  w = Num > 0;
  if (w)
     cout<<" Positive\n";
  }
```

`if/else` selection Structure


```
if ( Condition )
      statement;
else
      statement;
```

```
if ( Condition ){
      statement1;
      statement2;
      …
      }
else{
      statement1;
      statement2;
      …
}
```

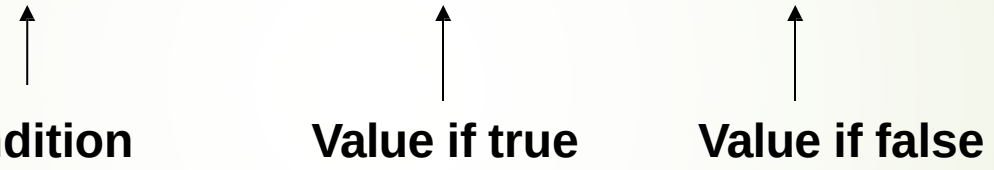
Read a mark, then print "PASS" if it is greater than or equal 50, or print "FAIL" otherwise

```
int main()
{
      int mark;

      cout<<"Please Enter your mark: ";
      cin >> mark;

      if (mark >= 50)
            cout<<" PASS\n";
      else
            cout<<"FAIL\n";
}
```

Ternary conditional operator

- Ternary conditional operator (`?:`)
  - Three arguments (condition, value if `true`, value if `false`)

```
cout <<( mark >= 50 ? "PASS\n" : "FAIL\n" );
```



- Equivalent to:

```
if (mark >= 50)
      cout<<" PASS\n";
else
      cout<<"FAIL\n";
```

More than one statement in **if**


```
int main(){

      int mark;
      cout << "Please Enter your mark: ";
      cin >> mark;

      if (mark >= 50){
            cout<<"PASS\n";
            cout<<"You can take the next course\n";
      }
      else {
         cout<<"FAIL\n";
         cout<<"You must take this course again\n";
      }
}
```

Write a program to print the fraction **a/b** in the form **c d/b**

```
int main(){
   int a,b,c,d;
   cout<<"To convert from a/b to c d/b, Enter a,b";
   cin >> a >> b;

   c = a / b;
   d = a % b;
   cout<< a << "/" << b << "=";

   if ( c != 0)
         cout<<c;
   if (d!=0)
   cout<<" "<<d<<"/"<<b;

}
```

Read any number, then print "positive" if it is positive and "negative" otherwise.

```
int main()
{

    int Num;
    cout<<"Please Enter Number:";
    cin>>Num;

    if (Num < 0)
        cout<<"Negative\n";
    else
        cout<<"Positive\n";
}
```

Equality (==) and Assignment (=) Operators


```
int main()
{

  int x = 0;

  if (x = 0)
    cout<<"condition is true\n";

  else
    cout<<"condition is false\n";
}
```

condition is false

Read two numbers and print the largest


```
int main(){

  int x,y;
  cout<<"Enter two numbers:";
  cin>>x>>y;
  cout<<"Max = ";

  if (x > y)
    cout<<x<<endl;
  else
    cout<<y<<endl;
}
```

```
Enter two numbers:15
4
Max = 15
```

Read three numbers and print the smallest


```
int main()
{

  int a, b, c;
  cout<<"Enter three numbers:\n";
  cin>>a>>b>>c;
  cout<<"Min = ";
  if ((a < b) && (a < c))
    cout<<a;
  if ((b < a) && (b < c))
    cout<<b;
  if ((c < a) && (c < b))
    cout<<c;
  }
```

```
Please Enter three numbers:
8
3
6
Min = 3
```

Read three numbers and print the smallest


```
int main(){
int a, b, c;
  cout<<"Please Enter three numbers:";
  cin>> a >> b >> c;
  cout<< "Min = ";
  int min = a;

  if (b < min)
  min = b;
  if (c < min)
  min = c;
  cout<<min;
}
```

```
Please Enter three numbers:8
3
6
Min = 3
```

Read three numbers and print the smallest, use nested **if**


```
int main(){
 int a, b, c;
     cout<<"Please Enter three numbers: ";
     cin>>a>>b>>c;
     cout<<"Min = ";
  if (a < b)
     if (a < c)
  cout<<a;
     else
          cout<<c;
  else if (b < c)
           cout<<b;
        else
           cout<<c;
 }
```

```
Please Enter three numbers: 5
11
9
Min = 5
```

Read a number, if it is positive add **10** to it and print Number "is positive", otherwise, subtract 10 and print Number "is negative"

```
int main(){
      int Number;
      cout<<"Please enter Number:";
      cin>>Number;

      if (Number>0) {
            Number = Number + 10;
            cout<<Number<<" is Positive\n";
      }

      else {
            Number = Number - 10;
            cout<<Number<<" is Negative\n";
      }
}
```

Dangling else


```
int main()
{
  int x = 2 , y = 5 , z = 10;
  if ( x > y)
    if ( x < z)
      cout <<" Hello";
    else
      cout <<"Hi";
}
```

Nothing is printed

Multiple Selection Structure (**switch**)

- Test variable for multiple values
- Series of **case** labels and optional **default** case

```
switch ( variable ) {
  case value1: // taken if variable = value1
    statements
    break;  // necessary to exit switch

  case value2:
  case value3: //taken if variable = value2 or = value3
   statements
    break;

  default: //taken if variable matches no other case
    statements
    break;
}
```

Example 1


```
int main(){
     int a;
     cout<<" Enter an Integer between 0 and 10: ";
     cin>>a;

     switch(a){
       case 0: case 1:  cout<<"hello ";
       case 2:   cout<<"there ";
       case 3:   cout<<"Welcome to ";
       case 4:   cout<<"C++ "<< endl;
              break;
       case 5:   cout<<"How ";
       case 6: case 7: case 8: cout<<"are you "<<endl;
              break;
       case 9:
              break;
       case 10:   cout<<"Have a nice day. "<<endl;
              break;
       default:   cout<<"the number is out of range"<<endl;
       }
     cout<< "Out of switch structure."<<endl;
}
```

Example 2


```
int main( ) {
 int score;
 char grade;
 cin >>score;
  switch(score/10)
  {
    case 0:case 1:case 2:case 3:case 4:case 5: grade='F';
    break;
    case 6: grade = 'D';        break;
    case 7: grade = 'C';        break;
    case 8: grade = 'B';         break;
    case 9: case 10: grade = 'A';   break;
    default: cout<<"Invalid test score."<<endl;
  }
   cout<<"Grade is"<<grade<<endl;
}
```

Example 3


```
int main( ) {

  char grade;
  cout <<" Enter grade as a letter : " ;
  cin>>grade;

  switch(grade){
    case   'A':    cout<<"The   Grade  is  A";   break;
    case   'B':    cout<<"The   Grade  is  B";   break;
    case   'C':    cout<<"The   Grade  is  C";   break;
    case   'D':    cout<<"The   Grade  is  D";   break;
    case   'F':    cout<<"The   Grade  is  F";   break;
   default: cout<< "The Grade is invalid";
    }
 }
```

Example 4


```
int main( )
{
  int age;
  cout<<"Enter your age: ";
  cin>>age;
  switch (age >= 18){
  case 1:
     cout<<"old    enough    to    drive"<<endl;
     cout<<"old enough to vote."<<endl;
     break;

  case 0:
     cout<<"Not old enough to drive"<<endl;
     cout<<"Not old enough to vote."<<endl;
  }
}
```

Quiz

- Write a program to read two numbers (a and b) and one character (op). The program then uses switch statement to print the output according to the table below:

op	output
+	a+b
-	a-b
*	a*b
/	a/b
otherwise	"Invalid Operation"

`for` Repetition Structure

- General format:

```
for ( initialization; condition; increment)
    statement;
```

- Statements will be executed repeatedly while condition is true.
- When the condition become false, the loop will be terminated and the execution sequence will go the first statement after for loop.
- If the loop body contains only one statement, there is no need to begin { and end } the loop body.

Examples


```
for( int c = 1; c <= 5; c++ )
     cout << c << endl;
```

```
1
2
3
4
5
```

```
for (int i = 1; i <= 5; i++)
          cout<<"Amman\n";
```

```
Amman
Amman
Amman
Amman
Amman
```

```
for (int i = 5; i >=1 ; i--)
          cout<<"Amman\n";
```

```
Amman
Amman
Amman
Amman
Amman
```

Print the following numbers:

1 3 5 7 9 11

```
for (int k=1; k<=11; k+=2)
        cout<<k<<"\t";
```

Print the following numbers

20 17 14 11 8 5 2

```
for (int m=20; m>=2; m-=3)
        cout<<m<<"\t";
```

Print the following numbers
1 2 3 4 … n (entered by user)

```
int main( )

{
     int n;
     cout<<"Enter the upper limit:";
     cin >> n;

     for (int i=1; i<=n; i++)
          cout<<i<<"\t";

     cout<<endl;
}
```

Print the following numbers
a (a+1) (a + 2) … b (a and b are entered by user)

```
int main( )
{

      int a,b;
      cout<<"Enter the start value:";
      cin>>a;

      cout<<"Enter the end value:";
      cin>>b;

      for (int i=a; i<=b; i++)
            cout<<i<<"\t";
}
```

Read five numbers from user and print the positive numbers only

```
int main( )
{
     int num;

     for (int i=1; i<=5; i++){
          cout<<"Please Enter No "<<i<<':';
          cin>>num;

          if (num > 0)
               cout<<num<<" is positive\n";
     }
 }
```

Compute and print S , Where S = 1 + 2 + 3 + 4 + 5


```
int S=0;
for (int i=1; i<=5; i++)
      S += i;
cout<<"Sum is "<<S<<endl;
```

Compute and print S, Where S = 1 + 3 + 5 + 7 + ⋯ + n

```
int Sum=0, n;  cout<<"Please
Enter n";  cin>>n;
for (int i=1; i<=n; i += 2)
      Sum += i;
cout<<"Sum="<<Sum<<endl;
```

Compute and print the summation of any 10 numbers entered by the user

```
int main()
{
   int S=0, N;
   for (int i = 10; i >= 1 ; i--)
   {
      cout<<"Enter the next number:";
      cin>>N;  S += N;
   }

      cout<<"Sum = "<< S <<endl;
}
```

Compute and Print the factorial of 5, where:

$$fact(5) = 5 \times 4 \times 3 \times 2 \times 1$$

```
int main( )
{

    int Fact=1;

    for (int j = 5; j >= 1; j--)
        Fact *= j;

    cout<<"5! = "<<Fact<<endl;
}
```

Compute and Print the factorial of n, where

$$fact(n) = n \times n-1 \times n-2 \times \cdots \times 1$$

```
int main( )
{
  int Fact = 1, n;

  cout<<"Enter an integer: ";
  cin>>n;

  for (int j=n; j>=1; j--)
       Fact *= j;

  cout<< n <<"! = "<<Fact<<endl;
}
```

Compute and Print the value of M where:

$$M = 2 \times 4 \times 6 \times \cdots \times n$$

```
int main( )
{

      long M = 1;
      int n;
      cout<<"please enter the upper Limit:";
      cin>>n;

      for (int i=2; i<=n; i += 2)
            M *= i;

      cout<<"M = "<< M <<endl;
}
```

Quiz

- Write a program that prints the numbers from X to Y, with step Z , using `for` statement. The program should read X, Y, Z then start the loop

Compute and Print M^n


```
int main()
{

      long Result = 1;
      int M, n;
      cout<<"Enter the Base number:";
      cin>> M;
      cout<<"Enter the exponent:";
      cin>> n;
      for (int i=1; i<=n; i++)
            Result *= M;

      cout<<"Result= "<<Result<<endl;
}
```

Quiz

- Write a program that finds M^n for positive and negative n

While Repetition Structure


```
initialization;
while (Condition)
{
    statements;
    increment;
}
```

- Statements will be executed repeatedly while condition is true
- When the condition become false, the loop will be terminated and the execution sequence will go to the first statement after While loop
- If the loop body contains only one statement, there is no need to begin { and end } the loop body.

Print the word "Amman" five times


```
int main( )
{
      int i=1;

       while (i<=5){
            cout<<"Amman\n";
            i++;
      }
}
```

```
Amman
Amman
Amman
Amman
Amman
```

Print the word "Amman" five times


```
int main()
{
  int i=1;

  while(i++ <= 5)
     cout<<"Amman\n";

  cout<<i<<endl;
}
```

```
Amman
Amman
Amman
Amman
Amman
7
```

Print the following numbers

1 3 5 7 9 11

```
int main()
{

    int i=1;

    while (i <= 11)
    {
        cout<<i<<'\t';
        i+=2;
    }
}
```

Print the following numbers

20 17 14 ... n

```
int main()
{
      int n, k=20;
      cout<<"Enter the lower limit:";
      cin>>n;

      while ( k >= n)
      {
            cout<<k<<'\t';
            k -= 3;
      }

      cout<<endl;
}
```

Read five numbers from the user and print the positive numbers only

```
int main()
 {
      int num, j=0;

      while ( j++ < 5 )
      {
            cout<<"Enter a number:";
            cin>>num;

            if (num > 0)
                  cout<<num<<endl;
      }
}
```

Sum of numbers from x to y


```
int main()
{
   int sum = 0, i, x, y;
   cout<<"Enter First Number: ";
   cin >> x;
   cout<<"Enter Second Number: ";
   cin >> y;

   i = x;
   while ( i <= y)
   {
       sum  =  sum  +  i;
       i = i+1;
   }
   cout<<"Sum from "<<x<<" to "<<y<<" = "<<sum;
}
```

```
Enter First Number: 5
Enter Second Number: 8
Sum from 5 to 8 = 26
```

Compute and print sum, Where $sum = 1 + 3 + 5 + 7 + \cdots + n$

```
int main()
{
      int n, Sum=0, i=1;
      cout<<"Enter the upper limit:";
      cin>>n;

      while ( i <= n )
      {
            Sum += i;
            i += 2;
      }

      cout<<"Sum="<<Sum<<endl;
}
```

Read 10 numbers and compute the sum of numbers divisible by 3


```
int main() {

      int Num, Sum=0, i=1;

      while ( i <= 10 ) {
            cout<<"Enter a number:";
            cin>>Num;

            if (Num % 3 == 0)
                  Sum += Num;
            i++;
      }

      cout<<"\nSum="<<Sum;
}
```

Compute and Print the value of M where:

$$M = 2 \times 4 \times 6 \times \cdots \times n$$

```
int main()
 {
      int N, M=1, i=2;
      cout<<"Enter the upper limit:";
      cin>>N;

      while ( i <= N ) {
              M *= i;
              i += 2;
      }

      cout<<"\nM="<<M;
}
```

Do While Repetition Structure


```
initialization
do {
      Statement(s);
} while (Condition) ;
```

- Statements will be executed repeatedly while condition is true
- When condition become false, the loop will be terminated and the execution sequence will go to the first statement after the loop
- The loop body will be executed at least once.

Print the word "Amman" five times


```
int main( ) {

    int i = 1;

    do {
        cout<<"Amman\n";
        i++;
    } while (i <= 5);

}
```

Program to read an integer then prints if it is *Even* or *Odd*. The program keeps running until number 1 is entered

```
int main() {
      int Choice, Num;
      do {
            cout <<"\nEnter a Number: ";
            cin >> Num;

            if (Num%2 == 0)
                  cout<<Num<<" is Even\n";
            else
                  cout<<Num<<" is Odd\n";

            cout<<"Enter 1 to Exit program\n";
            cout<<"Enter any other number to repeat\n";
            cin>>Choice;
      } while (Choice != 1); }
```

Modifying previous program such that 'Y' is entered to continue program and any other character to end


```
int main()
{
   int Num;
   char Choice;
   do {
       cout<<"\nEnter a Number: ";
       cin >> Num;

       if (Num%2 == 0)
              cout<<Num<<" is Even\n";
       else
              cout<<Num<<" is Odd\n";
       cout<<"Enter Y to continue\n";
       cout<<"Enter any other character to end program\n";
       cin>>Choice;
   } while (Choice == 'Y');
}
```

Modify previous program such that 'Y' or 'y' is entered to continue


```
int main() {
   int Num;
   char Choice;
   do {
       cout<<"\nEnter a Number";
       cin >> Num;
       if (Num%2 == 0)
             cout<<Num<<" is Even\n";
       else
             cout<<Num<<" is Odd\n";
       cout<<"Enter Y to continue\n";
       cout<<"Enter any other character to end
       program\n";
       cin>>Choice;
   } while ((Choice == 'Y') || (Choice =='y'));
}
```

`break` Statement

- Immediate exit from **`while`**, **`for`**, **`do/while`**, **`switch`**
- Program continues with first statement after structure
- Used to escape early from a loop
- Skip the remainder of **`switch`**

Example


```
int main ()
{
   int x;
   for (int i = 1; i <= 10; i++)
   {
      if (i == 5)
         break;
      cout<< x<< " ";
   }
   cout<<endl;

   cout<<"Broke out of loop when x became"<<x<<endl;
}
```

1 2 3 4

Broke out of loop when x became 5

Read a number and print "Prime" if it is a prime number, or "Not prime" otherwise

```
int main() {
  bool Prime = true;
  int i, num;
  cout<<"Please enter the number:";
  cin>>num;

  for ( i=2; i<num; i++)
    if (num%i==0) {
      Prime = false;
      break;
    }
  if (Prime)
    cout<<num<<" is a Prime number\n";
 else
    cout<<num<<" is not a Prime number\n";
}
```

`continue` Statements

- Used in `while`, `for`, `do/while`
- Skips remainder of loop body
- Proceeds with next iteration of loop

Example


```
int main(){
      for (int x = 1; x <= 10; x++)
      {
            if(x == 5)
                  continue;
            cout << x << " ";
      }

       cout<<endl;
      cout<<"skipped printing the value 5";
}
```

```
1 2 3 4 6 7 8 9 10
skipped printing the value 5
```

Read five numbers from user then print the positive numbers only (use **continue**)

```
int main( )
{
     int num;

     for (int i=1; i<=5; i++){
          cout<<"Please Enter No "<<i<<':';
          cin>>num;

          if (num < 0)
               continue;
     cout<<num<<" is positive\n";
     }
 }
```

Nested for

- **`for`** repetition structure that rests entirely within another **`for`** repetition structure

Outer loop ⟶

Inner loop ⟶

```
for(initialization; condition; increment)
    for(initialization; condition; increment)
        statement
```

- If the outer loop will repeat ***m*** times and the inner loop will repeat ***n*** times, then each statement in the inner loop will be executed ***m*** × ***n*** times

Nested for Example 1


```
for ( int i = 1; i <= 5 ;  i++)
{
  for (int j = 1 ; j <= 5 ; j++)
     cout << "*" ;
  cout<< endl;
}
```

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Nested for Example 2


```
int main()
{
  for(int i=1;i<=5;i++)
  {
     for (int j=1;j<=5;j++)
          cout<<i<<","<<j<<"  ";
     cout<<endl;
 }
}
```

1,1	1,2	1,3	1,4	1,5
2,1	2,2	2,3	2,4	2,5
3,1	3,2	3,3	3,4	3,5
4,1	4,2	4,3	4,4	4,5
5,1	5,2	5,3	5,4	5,5

Draw the following shape:


```
for (int r = 1 ; r <= 5; r++)
{
    for (int C = 1; C <= r; C++)
        cout<<'*';
    cout<<endl;
}
```

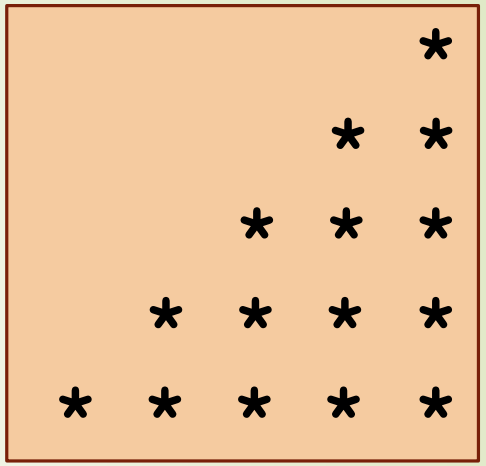
```
*
* *
* * *
* * * *
* * * * *
```

Draw the following shape:

```
* * * * *
* * * *
* * *
* *
*
```

```
for (int r = 1; r <=  5; r++)
 {
     for (int c = r; c <= 5; c++)
         cout<<'*';
     cout<<endl;
}
```

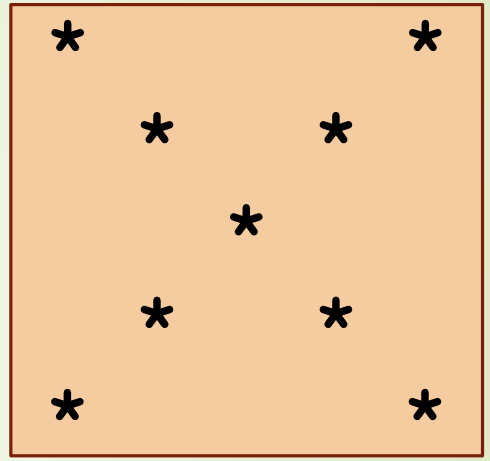
Draw the following shape:




```
for (int i = 1 ; i <= 5; i++)
{
    for(int k = i ; k < 5 ; k++)
        cout<<" ";
    for (int j = 1; j <= i; j++)
        cout<<'*';
    cout<<endl;
}
```

What is the output for the following program


```
for (int i = 1 ; i <= 5 ; i++)
{
   for (int j = 1; j <= 5; j++)
     if (i == j)
          cout<<"*";
     else if (i+j == 6)
            cout<<"*";
          else
            cout<<" ";
   cout<<endl;
}
```



Using nested **for**, display the multiplication table for the number 3

```
for   (int i=1;   i<=10; i++)
       cout<<"3  x "<<i<<" = "<<3*i<<endl;
```

calculate S, where $S = m^0 + m^1 + \cdots + m^n$


```
int main ()
{
int s=0, n, m, t;
cout<< "Enter m please :";
cin>> m;
cout<<"Enter n please :";

for (int i = 0 ; i <= n ; i++){
    t = 1;
    for (int j = 1 ; j <= i ; j++)
          t = t * m;
    s = s + t;
}
cout<<s<<endl;
}
```

Nested while


```
int main()
{
  int j = 1;
  while (j <= 4)
  {
     int i = 1;
     while(i <= 4){
          cout<<'*'<<"\t";
          i++;
     }
     j++;
     cout<<endl;
  }
}
```

```
*    *    *    *
*    *    *    *
*    *    *    *
*    *    *    *
```

Draw the following shape using nested while


```
int main()
{
      int i=1;
      while (i<=5)
      {
            int j=1;
            while (j<=i)
            {
                  cout<<'*';
                  j++;
            }
           cout<<endl;
           i++;
      }
}
```

```
*
* *
* * *
* * * *
* * * * *
```