Basic Data Types and Variables

Data types

- in programming languages define the type of data that a variable can hold.
- These types determine what kind of operations can be performed on the data and
- how much memory will be allocated for it.
- Here are the basic data types commonly found in most programming languages (like C++, Java, Python, etc.):

1. Integer (int):

   Used to store whole numbers (both positive and negative).

   e.g. int age = 25;

   range: Typically from -2,147,483,648 to 2,147,483,647 (varies by system).

Data types (Cont.…)

2. Floating-point (float, double):

   Used to store real numbers with decimal points.

   float: Less precision (usually up to 7 decimal places).

   e.g. float height = 5.9;

   double: More precision (up to 15 decimal places).

   e.g. double weight = 72.56789;

3. Character (char):

   Used to store a single character (letters, digits, symbols).

   e.g. char grade = 'A';

   Typically 1 byte, stores the ASCII value of the character.

Data types (Cont.…)

3. Boolean (bool):

   Used to store truth values true or false.

   e.g. bool isStudent = true;

4. String:

   Used to store sequences of characters (a collection of char).

   e.g. string name = “John";

Variables

- A **variable** is a container that stores data. You can assign a value to a variable and later use or modify it.

Declaring a Variable

syntax: <data_type> variable_name = value;

my suggestion:

- Use names that explain the purpose of the variable;

e.g. Instead of declaring the variable as **int x** (which holds the age), we can declare it as **int age**.

Variables

- Follow Naming Conventions

  in most of the programming languages, camelCase is common for variable names (start with a lowercase letter, and subsequent words are capitalized)

  e.g.

  ```
  int totalMarks;
  float averageHeight;
  bool isLoggedIn;
  ```

- Avoid Single Letter Names

  Except for variables with very small scopes (like in loops), avoid single-letter names like a, b, or x, as they don't convey meaning

  e.g. int p; instead you may use int producPrice;

Variables

- Avoid Abbreviations

  Abbreviations can be confusing. Instead, use complete words for better clarity.

  e.g. instead of int numStd;

  use int numberOfStudents;

- Avoid Keywords as Variable Name

  Don't use keywords or reserved words

  e.g. instead of int return;

  use int returnValue;

- Use Boolean Names that Suggest True/False Meaning

  When defining boolean variables, choose names that clearly indicate a true or false condition.

  e.g. bool isComplete;

  bool hasError;

  bool isStudent;

Variables

- Use Boolean Names that Suggest True/False Meaning

  When defining boolean variables, choose names that clearly indicate a true or false condition.

  e.g. bool isComplete;

  bool hasError;

  bool isStudent;

Assigning Operator

- is used in programming to assign values to variables
  - E.g.

int a = 5;

Left side
Name of Variable

Right side
Value / *expression*

Arithmetic

- Solve this expression

$$\frac{b^2 - 4ac}{2a}$$

Where a=4, b=25, and c=5

Manually = <Calculate yourself>

Try in c++

```
float returnValue = b*b - 4*a*c /2 *a
```

Do we get the correct answer?

Decision

If/else Condition

If condition is true

statements

If Ali's height is greater then 6 feet

Then

Ali can become a member of the Basket Ball team

If statement in C++

```
If (condition)
    statement ;
```

```
If (condition)
{
    statement 1;
    statement 2;
}
```

```
if (age1 > age2)
    cout<<“Student 1 is older than student 2” ;
```

Relational Operators

< less than

<= less than or equal to

== equal to

>= greater than or equal to

> greater than

!= not equal to

Relational Operators

```
a != b;
```

```
X = 0;
X == 0;
```

Example

If the student age is greater than 18 or his height is greater than five feet then put him on the foot balll team

Else

Put him on the chess team

Logical Operators

AND	&&
OR	||

Logical Operators

If a is greater than b
AND c is greater than d

In C++

```
if(a > b && c> d)
if(age > 18 || height > 5)
```

Make a small flow chart of this program and see the one to one correspondence of the chart and the code

Example

Code

```
if AmirAge  >  AmaraAge)
{
    cout<< “Amir is older than Amara” ;
}
else
{
    cout<<“Amir is younger than or of the same
    age as Amara”;
}
```

Flow Chart Symbols

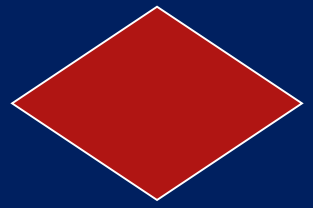
Start or stop

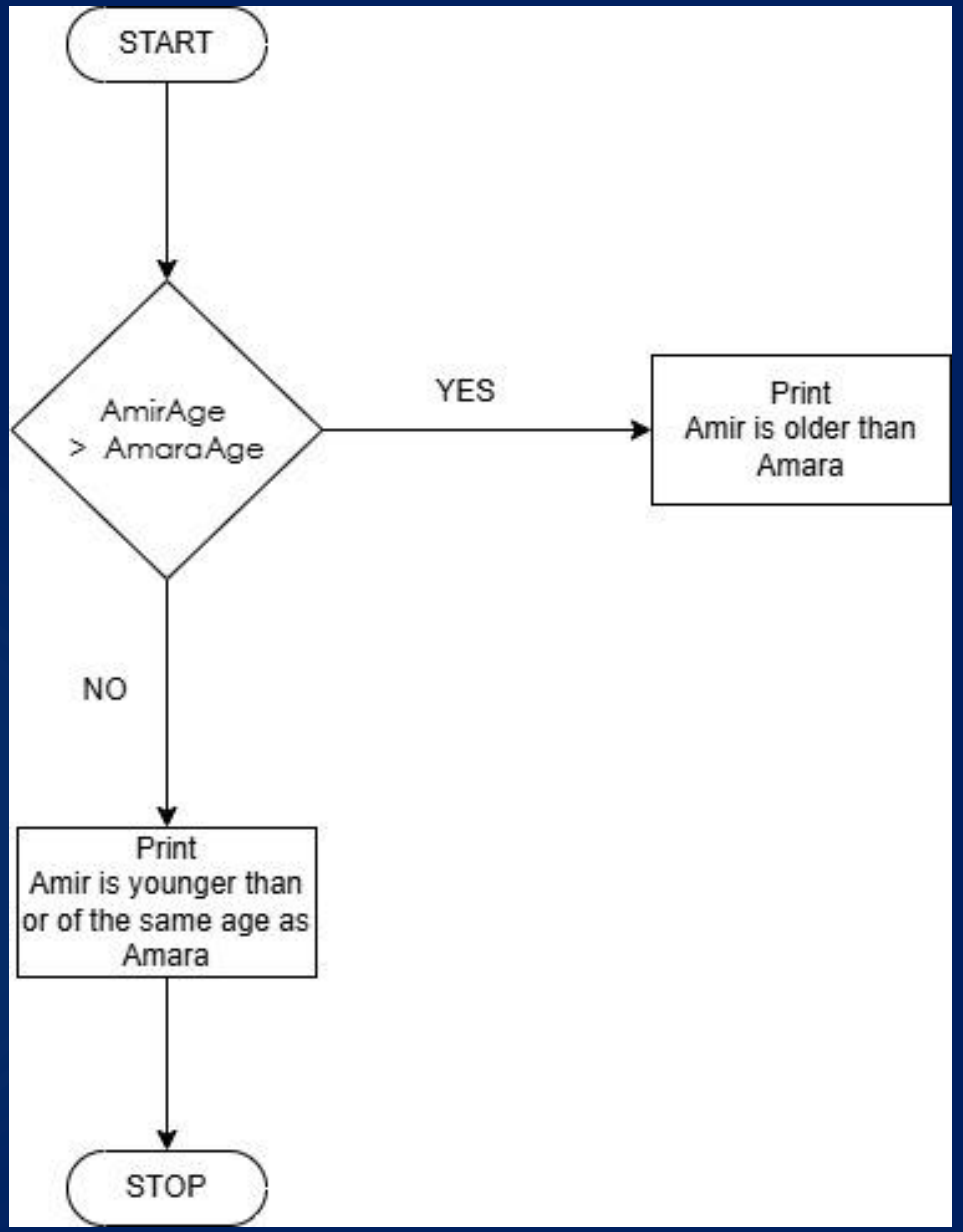
Process

Flow line

Continuation mark

Decision




START
AmirAge
> AmaraAge
YES
Print
Amir is older than
Amara
NO
Print
Amir is younger than
or of the same age as
Amara
STOP

Unary Not operator !

- !true = false
- !false = true

Example

```
if ((interMarks > 45) && (testMarks >= passMarks))
{
      cout << " Welcome to Newport Institute";
}
```

Example

If(!((interMarks > 45) && (testMarks >= passMarks)))

?

Unary Increment operator ++

count++

is same as

count=count + 1

Unary Decrement operator --

count--

is same as

count=count - 1

Nested if

```
If (age > 18)
{
    If(height  > 5)
    {
        cout << “you are in Foot Ball Team”<<endl;
    }
}
```

Make a flowchart of this nested if structure…