Classification of Programming Languages

- C++ Is a Middle-Level Language

Human language	English
• High level	python
• Middle level	C, C++
• Low level	Assembler
Machine language	binary

C++ Environment

- Compiler: is a software that converts source code written in a programming language like C++ to equivalent machine language in one go.

- IDE stands for an integrated development environment. IDE is a software application that provides facilities to a computer programmer for developing software.

STRUCTURE OF A C++ PROGRAM

```
int main() {                    // (main function)
     return 0;
}
```

* Main function is the starting point of running the program.
* (return 0) this line will return (0) at the end of executing the program to indicate that the program executed without any errors.
* ; at the end of the statement to indicate that the statement has ended

STRUCTURE OF A C++ PROGRAM

```
# include <iostream>
int main() {
      std::cout<<"This will be printed on the screen";
      return 0;
}
```

* #include is a way of including a standard or user-defined file in the program

* iostream is a header file that contains functions for basic input and output operations

The first line will include iostream header file to access input and output operations
* std means standard

* cout means character output

Standard character output

```
# include <iostream>
int main() {
      std::cout<<"This will be printed in the screen";
      std::cout<<"This also will be printed in the screen";
      std::cout<<"This also will be printed in the screen";
      return 0;
}
// screen output
This will be printed in the screenThis also will be printed in the screenThis also will be printed in the screen
```

Standard character output

```
# include <iostream>
int main() {
   std::cout<<"This is the first line"<<std::endl;
   std::cout<<"This is the second line"<<"\n";
   std::cout<<"This is the third line\n";
   std::cout<<"This is the fourth line";
   return 0;
}
```

std::endl and "\n" end the line, So the next print will start on a new line.

comments

- A comment is a text note that gives an explanation and remind programmers of what they did with the code.
- In C++, comments can be written in two ways:
- **a) Single line comments**: start with // (double slash) symbol.

```
// used for a single line
int a ; // declare variable a of an integer type
```

- **b) Multiple line comments**: Start with a /* symbol and end with a */ symbol.

```
/*  used for
a single
or
multiple
lines. */
```

comments

```
# include <iostream>
int main() {
    std::cout<<"This is the first line"<<std::endl;
    // std::cout<<"This line will not be displayed";
    std::cout<<"This is the second line"<<"\n";
    /*
    std::cout<<"These lines will ";
    std::cout<<"not be displayed";
    */
    std::cout<<"This is the third line\n";
    std::cout<<"This is the fourth line";
    return 0;
}
```

Identifier Names

- The names of variables, functions, labels, and various other user-defined objects are called identifiers. These identifiers can vary from one to several characters.
- The first character must be a **letter or an underscore**, and subsequent characters must be either **letters, digits, or underscores**.
- **Case sensitive**, i.e. uppercase and lowercase letters are different.
- Can’t use **reserved words**.

Identifier Names Examples

- **Correct** identifier names:

Count
test23
_high_balance

- **Incorrect** identifier names:

1count
hi!there
high...balance
int

Data Types:

- **1 byte = 8 bits**

Data type	Size in bytes	Description
int	2 or 4	Stores numbers, without decimals
float	4	Stores fractional numbers, with decimals
double	8	Stores fractional numbers, with decimals
bool	1	Stores true or false values
char	1	Stores single character/letter/number, or ASCII values.
std::string		Stores multiple characters, or words.

Variable Initializations

- data_type variable_name = value; // declared and assigned value

Or

- data_type variable_name; // declare the variable
- variable_name = value; // assign value to the variable

Variable Initializations Examples

- int my_num = 5; *// declared and assigned value*
- int my_num; *// declare the variable*
- my_num = 5; *// assign value to the variable*
- float my_floatNum = 5.99;
- float my_floatNum;
- my_floatNum = 5.99;
- double my_doubleNum = 9.98;
- char my_letter = 'D';
- bool my_boolean = true;

```cpp
# include <iostream>
int main() {
    // char
    char c = 'a';
    std::cout << c << std::endl;

    // string
    std::string x = "these are multiple words";
    std::cout << x << "\n";

    // int
    int i = 3;
    std::cout << i << std::endl;
    printf("this is c command to print i: %d\n", i); // printf is c command not c++

    // float
    float a,b;
    a = 4.6 ;
    b = 11 ;
    std::cout<<"this is a: "<< a <<", and this is b: "<<b<<"\n";

    // double
    double ss = 5;
    std::cout<< ss <<std::endl;
    ss = 44;        // assign another value for ss
    std::cout<< ss <<std::endl;

    // boolean
    bool z = false;
    std::cout<<z<<std::endl;

    return 0;
}
```

Constant Variable

- A **constant variable**: is one whose value cannot be updated or altered anywhere in the program. A constant variable must be initialized at its declaration. (read-only variable)
- Common naming convention for constants is making all letters in uppercase.

```
#include <iostream>
int main() {
    double pi = 3.14;   // double variable (pi) initialized and assigned value
    pi = 33;            // reassign value to pi.  If pi is const, we can't reassign it.
    double radius = 5;
    double circumference = 2 * pi * radius;
    std::cout << circumference << "cm\n";
    return 0;
}
```

```
    const double PI = 3.14;
    double radius = 5;
    double circumference = 2 * PI * radius;
    std::cout << circumference << "cm\n";
```

Namespace

- A namespace is a declarative region that provides a scope to the identifiers inside it. Namespaces are used to organize code into logical groups and to prevent name collisions that can occur especially when the code base includes multiple libraries.

```
#include <iostream>
namespace first{
int x = 10;}
namespace second{
int x = 20;}
int main() {
    int x = 5;
    std::cout << x << "\n";
    std::cout << first::x << "\n";
    std::cout << second::x << "\n";
    return 0;
}
```

Namespace

```cpp
#include <iostream>
namespace first{
int x = 10;}
namespace second{
int x = 20;}
int main() {
   using namespace second;   // here we used namespace second in main function
   std::cout << x << "\n";
   std::cout << first::x << "\n";
   std::cout << second::x << "\n";
   return 0;
}
```

Namespace

```
#include <iostream>
int main() {
   using namespace std;
   string x = "these are multiple words";
   cout << x << "\n";
   return 0;
}
```

Arithmetic Operators

- Arithmetic operators are used to perform operations on variables and values.

Operator	Name of the operator
+	Addition
-	Subtraction
*	multiplication
/	division
%	modulus

+ operator (addition)

```
int x = 25 + 50;                    // 75 (25 + 50)
int sum1 = 100 + 50;                // 150 (100 + 50)
int sum2 = sum1 + 250;              // 400 (150 + 250)
int sum3 = sum2 + sum2;             // 800 (400 + 400)
sum3 += 1 ;                         // 801 (800 + 1 )
sum3 += 2 ;                         // 803 (801 + 2 )
sum3 ++ ;                           // 804 (803 + 1 )
std::cout << 3 + 5 << "\n";         // 8
```

- operator (subtraction)

```
int x = 25 - 50;                    // -25 (25 - 50)
int sub1 = 100 - 50;                // 50 (100 - 50)
int sub2 = sub1 - 250;              // -200 (50 - 250)
int sub3 = sub2 - sub2;             // 0 (-200 - -200)
sub3 -= 1 ;                         // -1 (00 - 1 )
sub3 -= 10 ;                        // -3 (-1 - 2 )
sub3 -- ;                           // -4 (-3 - 1 )
std::cout << 3 - 5 << "\n";         // -2
```

* operator (multiplication)

```
int x = 25 * 50;                    // 1250 (25 * 50)
int mul1 = 3 * 2;                   // 6 (3 * 2)
int mul2 = mul1 * 2;                // 12 (6 * 2)
int mul3 = mul2 * mul2;             // 144 (12 * 12)
mul3 *= 1 ;                         // 144 (144 * 1 )
mul3 *= 2 ;                         // 288 (144 * 2 )
std::cout << 3 * 5 << "\n";         // 15
```

/ operator (division)

```
int x = 50 / 25;                        // 2 (50 / 25)
int div1 = 12 / 2;                      // 6 (12 / 2)
int div2 = div1 / 2;                    // 3 (6 / 2)
int div3 = div2 / div2;                 // 1 (3 / 3)
div1 /= 1 ;                             // 6 (6 / 1 )
div1 /= 2 ;                             // 3 (6 / 2 )
std::cout << 10 / 5 << "\n";            // 2
int y = 9 / 5 ;                         // 1
std::cout << y << "\n";
float z = 9 / 5 ;                       // 1
std::cout << z << "\n";
float z = 9.1 / 5 ;                     // 1.82
float x = 5
float z;
z = x / 2;                              // 2.5
```

% operator (modulus)

```
int x = 5 % 2; // 1 (5 % 2)
int div1 = 14 % 5; // 4 (14 % 5)
int div2 = 12 % 4; // 0 (12 % 4)
int div3 = 10 % 10; // 0 (10 % 10)
div1 %= 1 ; // 0 (4 % 1 )
div1 %= 3 ; // 1 (4 % 3 )
std::cout << 10 % 4 << "\n"; // 2
float z = 9 % 5 ;
std::cout << z << "\n";
```

Arithmetic operators' precedence

1- parenthesis
2- multiplication and division
3- addition and subtraction
4- left to right

4 - 5 + 2 * 6 / (3 + 1)
4 - 5 + 2 * 6 / 4 // (3 + 1) = 4
4 - 5 + 3 // 2 * 6 / 4 = 3
2 // 4 - 5 + 3 = 2

(14 – 7) * 2 + 6 / 3 + 1

7 * 2 + 6 / 3 + 1 // (14 – 7) = 7

14 + 2 + 1 // 7 * 2 = 14, 6 / 3 = 2

17 // 14 + 2 + 1 = 17

Standard character input

This is used to take input into the program.

```
int age ;
std::cout << "enter your age: ";
std::cin >> age ;   // this will prombt the user to enter his age
std::cout << "Your age is " << age << "\n";
```

* std: standard namespace
* cin: character input

```cpp
#include<iostream>
int main() {
std::string name;
int age;
std::cout << "enter your name: ";
std::cin >> name;
std::cout << "enter your age: ";
std::cin >> age;
std::cout << "Your name is: " << name << ", your age is " << age << "\n";
return 0;
}
```

Comparison operators

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

if statement

- if the condition is true then execute the code.

```
#include<iostream>
int main() {
   int age = 7;
   if (age >= 6){ // condition is true if age equals or is older than 6
      std::cout << "welcome to school\n";
   }
   return 0;
}
```

if else statement

- if the condition is true then execute the code, but if the condition is false execute another code.

```
#include<iostream>
int main() {
    int age = 4;
    if (age >= 6){ // condition is true if age equals or is older than 6
        std::cout << "welcome to school\n";
    } else {
        std::cout << "you are not old enough to enter school\n";
    }
    return 0;
}
```

```
#include<iostream>
int main() {
    int age = 5;
    if (age < 0){
        std::cout << ”this is not a correct age\n";
    } else if (age >= 6){
        std::cout << "welcome to school\n";
    } else {
        std::cout << "you are not old enough to enter school\n";
    }
    return 0;
}
```

Logical Operators

- && : AND operator
- || : OR operator
- ! : NOT operator

```
std::string username;
std::string password;
std::cout << "enter your username: ";
std::cin >> username;
std::cout << "enter your password: ";
std::cin >> password;
if (username == "abc" && password == "xyz"){
std::cout << "you can login\n";
} else {
std::cout << "you can't login\n";
}
```

```
std::string username;
std::string password;
std::cout << "enter your username: ";
std::cin >> username;
std::cout << "enter your password: ";
std::cin >> password;
if (username != "abc" || password != "xyz"){
std::cout << "you can't login\n";
} else {
std::cout << "you can login\n";
}
```

```
std::string username;
std::string password;
bool agree;
std::cout << "do you accept the terms and conditions? ";
std::cin >> agree;
std::cout << "enter your username: ";
std::cin >> username;
std::cout << "enter your password: ";
std::cin >> password;
if (!agree || username != "abc" || password != "xyz"){
std::cout << "you can't login\n";
} else {
std::cout << "you can login\n";
}
return 0;
```

```
std::string username;
std::string password;
bool agree;
std::cout << "do you accept the terms and conditions? ";
std::cin >> agree;
if (!agree) {
std::cout << "you didn’t accept the terms and conditions.\n";
return 0;
}
std::cout << "enter your username: ";
std::cin >> username;
std::cout << "enter your password: ";
std::cin >> password;
if (username == "abc" && password == "xyz"){
std::cout << "you can login\n";
} else {
std::cout << "you can't login\n";
}
return 0;
```

```cpp
#include <iostream>
using namespace std;
int main() {
    int mark;
    cout << "enter student mark:";
    cin >> mark;
    if (mark >= 0 && mark <=100){
        if (mark < 50) {cout << "fail\n";}
        else if (mark >= 50 && mark < 70) {cout << "pass\n";}
        else if (mark >= 70 && mark < 80) {cout << "good\n";}
        else if (mark >= 80 && mark < 90) {cout << "very good\n";}
        else if (mark >= 90 && mark <= 100) {cout << "excellent\n";}
    } else {cout<<"wrong number";}
    return 0;
}
```

Switch statement

- It is a cleaner way of using nested if statement. It will check one variable with match cases.

```
#include <iostream>
int main() {
    int month;
    std::cout << "Enter the month (1-12): ";
    std:: cin >> month;
    if(month == 1){std:: cout << "It is January";}
    else if(month == 2){std:: cout << "It is February";}
    else if(month == 3){std:: cout << "It is March";}
    else if(month == 4){std::cout << "It is April";}
    else if(month == 5){std::cout << "It is May";}
    else if(month == 6){std:: cout << "It is June";}
    else if(month == 7){std::cout << "It is July";}
    else if(month == 8){std::cout << "It is August";}
    else if(month == 9){std::cout << "It is September";}
    else if(month == 10){std::cout << "It is October";}
    else if(month == 11){std:: cout << "It is November";}
    else if(month == 12){std:: cout << "It is December";}
    else{std:: cout << "You didn't enter a number between (1-12)";}
    return 0;
}
```

```cpp
#include <iostream>
int main() {
    int month;
    std::cout << "Enter the month (1-12): ";
    std::cin >> month;
    switch(month) {
        case 1:
            std:: cout << "It is January";
            break;
        case 2:
            std:: cout << "It is February";
            break;
        case 3:
            std:: cout << "It is March";
            break;
        case 4:
            std:: cout << "It is April";
            break;
        case 5:
            std:: cout << "It is May";
            break;
        case 6:
            std:: cout << "It is June";
            break;
```

```
case 7:
    std:: cout << "It is July";
    break;
case 8:
    std:: cout << "It is August";
    break;
case 9:
    std:: cout << "It is September";
    break;
case 10:
    std:: cout << "It is October";
    break;
case 11:
    std:: cout << "It is November";
    break;
case 12:
    std:: cout << "It is December";
    break;
default:
    std:: cout << "You didn't enter a number between (1-12)";
}
return 0;
}
```

- in this program if the user enter A, or B it will show “morning boys class”, if the user enter C it will show “morning girls class”, and if the user enter D it will show “evening class”

```
#include <iostream>
using namespace std;
int main() {
    char studentClass;
    cout << "Enter student's class: ";
    cin >> studentClass;
    switch(studentClass) {
        case 'A’:
        case 'B':
            cout << "morning boys class";
            break;
        case 'C':
            cout << "morning girls class";
            break;
        case 'D':
            cout << "evening class";
            break;
        default:
            cout << "wrong class";
    }
    return 0;
}
```

While loop

- It is like if statement, except we repeat checking the condition if it is true we rerun the statement until the condition is false.

```
# include <iostream>
using namespace std;
int main(){
   string name;
   if (name.empty()) {
      cout << "Enter your name: ";
      getline(cin, name);
   }
   cout << "Hello " << name << "\n";
   return 0;
}
```

While loop

```
# include <iostream>
using namespace std;
int main() {
   string name;
   while (name == "") {
      cout << "Enter your name: ";
      getline(cin, name);
   }
   cout << "Hello " << name << "\n";
   return 0;
}
```

Do while loop

- In do-while loop it run the statement then check the condition if it is true will run the statement again and so on.

```
# include <iostream>
using namespace std;
int main() {
    int number;
    do {
        cout << "Enter a positive number: ";
        cin >> number;
    }while(number < 0);
    cout << "The number is: " << number << "\n";
    return 0;
}
```

For loop

- It executes a block of code a specified amount of times.

```
# include <iostream>
using namespace std;
int main() {
    for (int i=0; i < 4; i++) {
    cout << i << "\n";
    }
    cout<<"this is after for loop\n";
    return 0;
}
```

- `int i=0` represents the starting index.
- `i < 4` represents the stop condition.
- `i++` represents the increment amount for index.

- We can also decrement the index

```
for (int i=10; i > 4; i--) {
cout << i << "\n";
}
```

- We can increment or decrement by more than 1.

```
for (int i=10; i > 4; i-=3)
for (int i=0; i < 4; i+=2)
```

Nested for loop

```
# include <iostream>
using namespace std;
int main() {
   for (int i=0; i < 4; i++) {
      for (int j=0; j < 6; j+=2)
          cout << i << "," << j << "\n";
      }
   return 0;
}
```

Break and Continue in for loop

- Break: will break out of the loop.

```
# include <iostream>
using namespace std;
int main() {
   for (int i=0; i < 10; i++) {
      if (i == 4) break;
      cout << i << ” ";
   }
   return 0;
}
// the output will be 0 1 2 3
```

Break and Continue in for loop

- Continue: will just skip the current iteration.

```
# include <iostream>
using namespace std;
int main() {
    for (int i=0; i < 10; i++) {
        if (i == 4) continue;
        cout << i << " ";
    }
    return 0;
}
// the output will be 0 1 2 3 5 6 7 8 9
// without number 4
```

```
# include <iostream>
using namespace std;
int main() {
   for (int i=1; i <= 3; i++) {
      for (int j=1; j <= 6; j++){
          cout << "* ";
      }
      cout << "\n";
   }
   return 0;
}
// the output will be
* * * * * *
* * * * * *
* * * * * *
```

```cpp
# include <iostream>
using namespace std;
int main() {
   for (int i=1; i <= 5; i++) {
      for (int j=1; j <= i; j++){
          cout << "* ";
      }
      cout << "\n";
   }
   return 0;
}
// the output will be
*
* *
* * *
* * * *
* * * * *
```

```cpp
# include <iostream>
using namespace std;
int main() {
   for (int i=1; i <= 5; i++) {
      for (int j=1; j <= i; j++){
          cout << j << " ";
      }
      cout << "\n";
   }
   return 0;
}
// the output will be
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
# include <iostream>
using namespace std;
int main() {
    for (int i=0; i < 3; i++) {
        for (int j=0; j < 3; j++){
            if(j==1 && i==1)
                cout<< "  ";
            else cout << "* ";
        }
        cout << "\n";
    }
    return 0;
}
// the output will be
* * *
*   *
* * *
```

Array

- Array is a sequential list of values. You can use arrays to store and access multiple values of the same data type under one identifier.

```
#include<iostream>
using namespace std;
int main() {
   double x[3];
   x[0] = 19.2;
   x[1] = 7.9;
   x[2] = 54;
   cout<< x[1];
   return 0;
}
```

```cpp
#include<iostream>
using namespace std;
int main() {
    int x[3] = {19, 7, 54};
    int y[] = {1, 73, 42, 67, 2};
    int z[8] = {42, 67, 2};
    z[6] = 2;
    cout<<"write a number: ";
    cin >> z[7];
    cout<<"write 3 numbers: ";
    for (int i = 3; i < 6; ++i) {
        cin >> z[i];
    }
    cout<< x[1] << "\n";
    for (int i = 0; i < 8; ++i) {
        cout << z[i] << " ";
    }
    return 0;
}
```