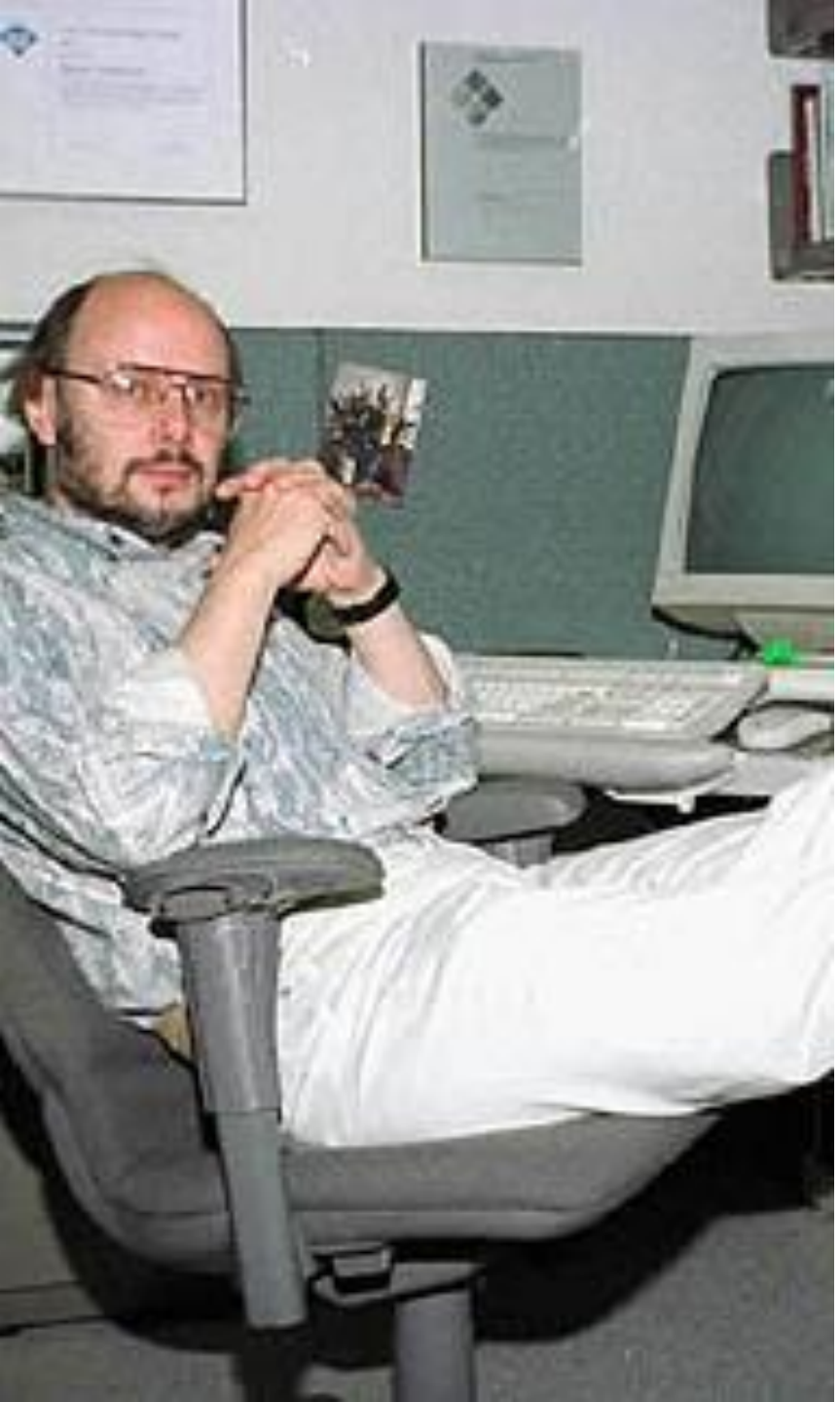
Introduction to C and C++

The C Language

- C was first developed by Dennis Ritchie and Bell Labs in 1972.
- Later formally published with the book “The C Programming Language” with Brian Kernighan in 1978.
- C contained primitive variables, functions, standard input and output stream, structures. The paradigm was procedural (before OO).



The C++ Language

- C++ was developed by Bjarne Stroustroup and his team at Bell Labs throughout the 1980s.
- The language was first referred to as ‘C with classes’, as it sought to apply the object-oriented paradigm to the original C language.
- The increment operator (++) increases the integer value of a variable by one. This indicates that C++ is the next increment of the original C language.

PC specifications 1983 vs 2023

	1983	2023
CPU		64 bit (mostly) Intel i9 AMD Ryzen Apple's M2
Clock speed		3.5GHz – 5Ghz
RAM		8gb - 32gb (some 64gb+)
Storage		SSDs: 500gb – 2tb

PC specifications 1983 vs 2023

	1983	2023
CPU	8 bit (some 16 bit) Intel 8088 Motorola 68000	64 bit (mostly) Intel i9 AMD Ryzen Apple's M2
Clock speed		3.5GHz – 5Ghz
RAM		8gb - 32gb (some 64gb+)
Storage		SSDs: 500gb – 2tb

PC specifications 1983 vs 2023

	1983	2023
CPU	8 bit (some 16 bit) Intel 8088 Motorola 68000	64 bit (mostly) Intel i9 AMD Ryzen Apple's M2
Clock speed	A few MHz (some up to 16MHz)	3.5GHz – 5Ghz
RAM		8gb - 32gb (some 64gb+)
Storage		SSDs: 500gb – 2tb

PC specifications 1983 vs 2023

	1983	2023
CPU	8 bit (some 16 bit) Intel 8088 Motorola 68000	64 bit (mostly) Intel i9 AMD Ryzen Apple's M2
Clock speed	A few MHz (some up to 16MHz)	3.5GHz – 5Ghz
RAM	64kb - 640kb	8gb - 32gb (some 64gb+)
Storage		SSDs: 500gb – 2tb

PC specifications 1983 vs 2023

	1983	2023
CPU	8 bit (some 16 bit) Intel 8088 Motorola 68000	64 bit (mostly) Intel i9 AMD Ryzen Apple's M2
Clock speed	A few MHz (some up to 16MHz)	3.5GHz – 5Ghz
RAM	64kb - 640kb	8gb - 32gb (some 64gb+)
Storage	IBM Personal Computer XT introduced a 10mb hdd in 1983	SSDs: 500gb – 2tb

Introduction to C

The original C language is renowned for having these features:

- Universally usable modular programs
- Efficient, close to machine programming
- Portable programs for various platforms.

Introduction to C++

C++ was developed by Bjarne Stroustrup and his team at Bell Labs throughout the 1980s. The language was first referred to as ‘C with classes’, as it sought to apply the object oriented paradigm to the original C language.

The increment operator (++) increases the integer value of a variable by one. This indicates that C++ is the next increment of the original C language. C# (2001) gets its name from music; in music theory, the note C# is one semitone higher in pitch than the note C.

Introduction to C++

C++ is not ‘purely’ object-oriented but draws on the paradigm (OOP) as well as the procedural paradigm (C). The benefit of this, is that existing C code can also be in C++ programs.

C++ supports the main concepts of object-oriented programming:

- **Data abstraction**: creation of classes to describe objects
- **Data encapsulation**: controlled access to object data
- **Inheritance**: creating derived classes (multiple derived classes)
- **Polymorphism** (Greek for many forms): implementation of instructions can have varying effects during program execution.

Compilation vs Interpretation

Compilation and execution

Modern applications are coded in "high-level” languages that provide a ‘high-level’ of abstraction between machine code, and source code that is written and read by human beings.

In order to run programs, source code must first be converted into machine code that the computer can understand and execute.

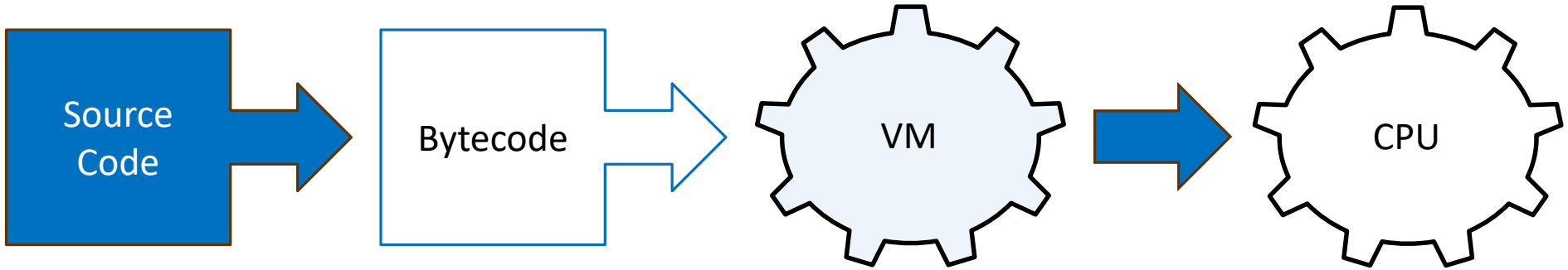
This process is accomplished either by an “interpreter” or by a “compiler” depending on which language the program is written.

Python uses an interpreter

The Python programming language uses an interpreter to translate program source code. As a program proceeds, the interpreter dynamically translates its functions and statement code objects into ”bytecode”. The bytecode can then be executed via Python’s bytecode interpreter (Virtual Machine / VM).

Java and Ruby also employ an interpreter to translate source code into bytecode for execution by the computer via their virtual machine.

Python Interpretation

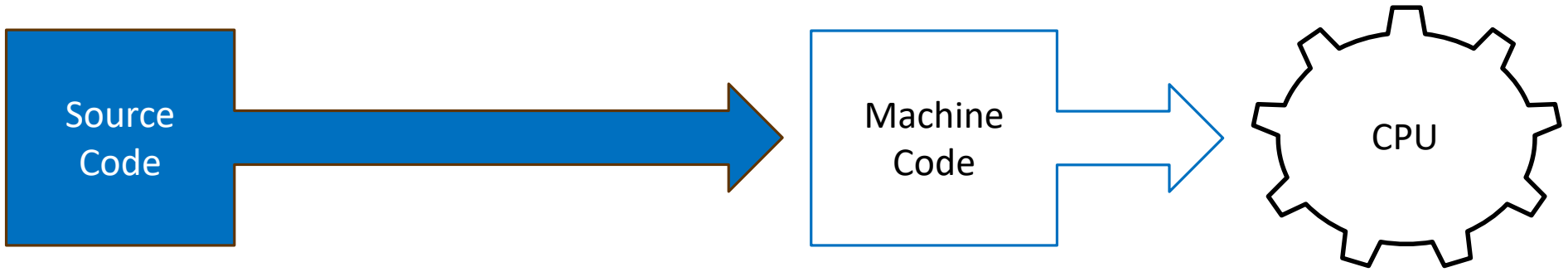


C uses a compiler

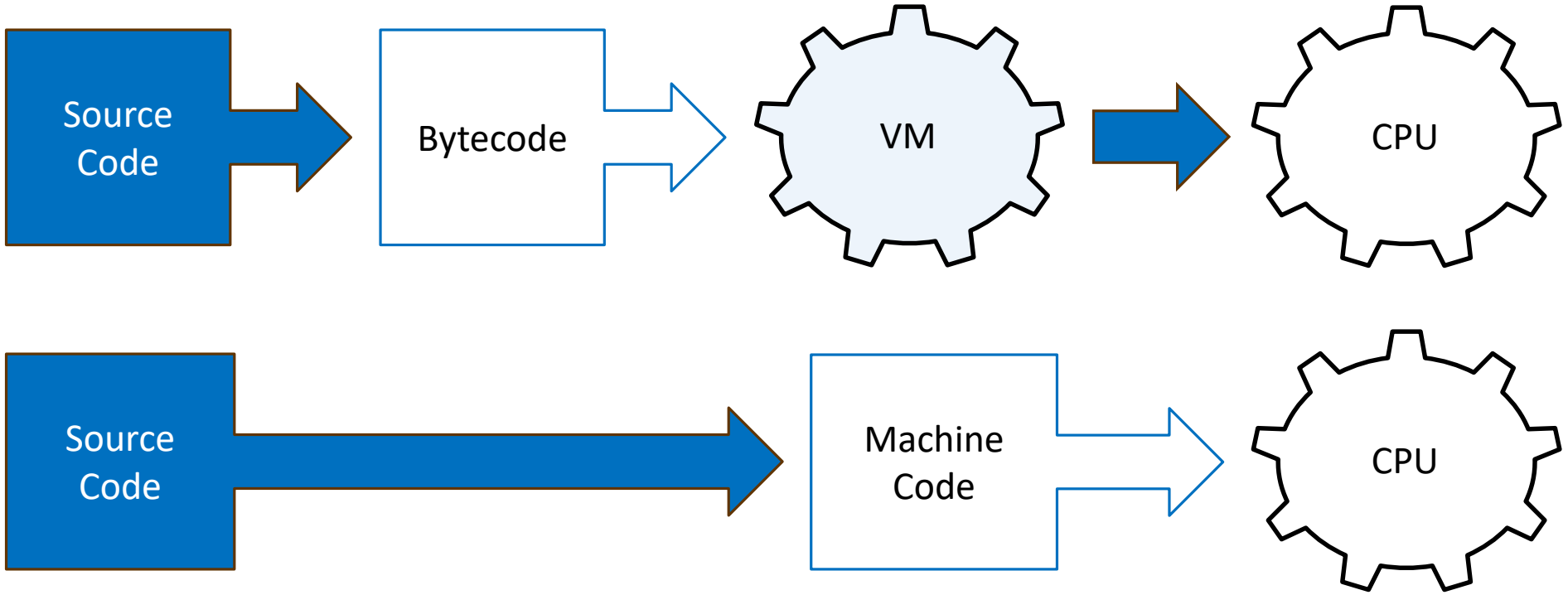
C, C++ and C# use a compiler to convert source code into machine code. Compilation takes the entire source code and first generates intermediate “object code” representing the program’s functions and statements in binary machine code format.

The compiler then combines the object code into a single binary machine code file (.exe) that can be executed directly by the computer.

Compilation in C



Comparison: Python and C



Interpreter vs Compiler

Interpreter	Compiler
Takes individual code objects as input for translation	Takes the entire source code as input for conversion
Does not generate intermediate object code	Does generate intermediate object code
Executes flow control statements slowly	Executes flow control statements quickly
Requires little memory as object code is not generated	Requires more memory as object code is generated

Interpreter vs Compiler

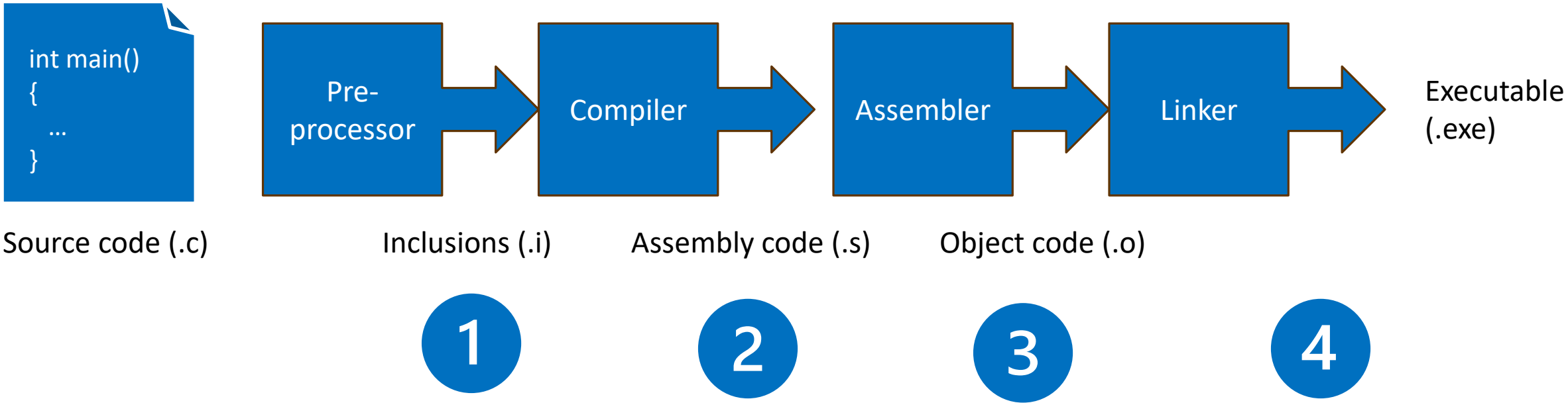
Interpreter	Compiler
Reports any errors immediately when they occur in a statement	Reports any errors only after an attempt to compile the entire source code
Programs run slower while code objects get individually translated to machine code	Programs run faster while machine code runs directly on the computer
Distributed programs are human-readable source code so are easy to modify	Distributed programs are compiled machine code so are difficult to modify
Offers poorer protection of intellectual property rights	Offers better protection of intellectual property rights

Four stages of C compilation

To produce an executable file from an original C source code file, the compilation process undergoes four separate stages:

1. Pre-processing
2. Translating
3. Assembling
4. Linking

Compiling C code



The C language vs Python

Exploration of the differences

In the following section we shall explore some of the differences between the C/C++ and Python language.

It is not intended to be an exhaustive coverage of all C/C++ language features, but highlight some of the key differences in approach.

The intention is to also reinforce the core programming concepts. Same concepts, different syntax/expression across language.

Output

Output (Hello World) in C vs Python


```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

```python
print("Hello, World!")
```

Output (Hello World) in C++ vs Python

```cpp
#include <iostream>

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

```python
print("Hello, World!")
```

Output (Hello World) in C++ vs Python


```cpp
#include <iostream>

using namespace std;

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

```python
print("Hello, World!")
```

C Namespaces

```
namespace firstNamespace{
        int a = 10;
}

namespace secondNamespace{
        int a = 12;
}

int main(){
        using firstNamespace::a
        int c = a + secondNamespace::a;
        std::cout << c << endl;
}
```

Input

Input in C vs Python

```c
#include <stdio.h>

int main() {
    int number;
    printf("Enter an integer: ");
    scanf("%d", &number);
    printf("You entered: %d\n", number);
    return 0;
}
```


```python
number = int(input("Enter an integer: "))
print("You entered:", number)
```

Input in C++ vs Python

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

int main() {
    int number;
    cin >> number;
    cout << "You entered" << number << endl;
    return 0;
}
```

```python
number = int(input("Enter an integer
print("You entered:", number)
```

Input in C++ vs Python

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

int main() {
    string input;
    cin >> input;
    cout << "You entered" << input << endl;
    return 0;
}
```

```python
input = input("Enter an integer: ")
print("You entered:", input)
```

Variables and Types

C Data types

C/C++ is strongly typed. Each variable must define the type of data being stored and an identifier (which holds the memory address for this value). Common primitive data types are:

Name	Description	Size	Range
char	Character or small int	1 byte	-128 to 127
int	Integer	2 bytes (16 bit)	-32768 to 32767
bool	Boolean value	1 byte	true or false
float	Floating point number	4 bytes	-3.4×10^{38} to 3.4×10^{38}

Variable names

Identifier names must follow these rules:

- Must begin with a letter or underscore, and cannot have spaces
- Must not be a reserved word
- Case sensitive
- Best practice is to choose a meaningful name!

```
char grade;
int age;
bool enrolled;
float height;
```

Constants

Unlike variables, constants are intended for values which do not ‘vary’ or ‘change’ throughout the lifetime of the program.

Constants can be defined with the prefix **const** keyword (which is the equivalent of **final** in Java).

```
const int HOURS_IN_DAY = 24;

const int MAX_MARK = 100;

const int ADULT_AGE = 18;
```

Selection

If statements in C vs Python

```c
#include <stdio.h>

int main() {
    int number = 5;
    if (number > 0){
        printf("The number is positive.\n");
    }
    return 0;
}
```

```python
number = 5
if number > 0:
    print("The number is positive.")
```

If statements in C vs Python

```c
int main() {
    int number = 5;
    if (number > 0){
        printf("The number is positive.\n");
    }
    else if(number < 0){
        printf("The number is negative.\n");
    }
    else{
        printf("The number is 0.\n");
    }
    return 0;
```

```python
number = 5
if number > 0:
    print("The number is positive.")
elif (number < 0):
    print("The number is negative.")
else:
    print("The number is 0.")
```

switch statement

The switch statement selects between cases

```
char grade = 'C';

switch(grade) {
    case 'A' : cout << "You achieved an A grade" << endl; break;
    case 'B' : cout << "You achieved a B grade" << endl; break;
    case 'C' : cout << "You achieved a C grade" << endl; break;
    case 'D' : cout << "You achieved a D grade" << endl; break;
    case 'F' : cout << "You failed this course" << endl; break;
    default : cout << "Invalid grade" << endl;
}
```

Ternary operator / arithmetic if

The arithmetic if operator takes three arguments and is a more condensed way of selecting between a true and false option.

```
int a = 18;
int b = 7;
int highest;

highest = (a > b)   ?    a   :    b;    //assign highest

cout << "The highest number is " << highest;
```

Iteration

For loops in C vs Python


```c
#include <stdio.h>

int main() {
    for (int i = 0; i <= 5; i++){
        printf("%d\n", i);
    }
    return 0;
}
```

```python
for i in range(1,6):
    print(i)
```

foreach in C++

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

int main() {
    int numbers[5] = {1,2,3,4,5};
    for (int i : numbers){
        cout << numbers[i] << endl;
    }
    return 0;
}
```

```python
for i in range(1,6):
    print(i)
```

C++ do while loop

The do while loop repeats whilst true, but the comparison takes place after one iteration of the loop. So loop repeats at least once.

```
char response;
do
 {
     // program instructions go here
     cout << "another go (y/n)?" << endl;
     cin >> response;
 }
 while (response == 'y');
```

Functions

Functions in C vs Python

```c
#include <stdio.h>

void say_hello();  //declaration

int main() {
    say_hello();
    return 0;
}

void say_hello(){    //definition
     printf("Hello, World!\n");
}
```

```python
def say_hello():
    print("Hello, World!")

say_hello()
```

Arrays

Arrays in C vs Python (List)

```c
#include <stdio.h>

int main() {
    int myArray[5] = {1, 2, 3, 4, 5};

    for (int i = 0; i < 5; i++) {
        printf("%d ", myArray[i]);
    }
    return 0;
}
```

```python
myList = [1,2,3,4,5]

print(myList)
```

C Arrays vs Python Lists

```c
#include <stdio.h>

int main() {
    int myArray[5] = {1, 2, 3, 4, 5};

    for (int i = 0; i < 5; i++) {
        printf("%d ", myArray[i]);
    }
    return 0;
}
```

```python
myList = [1,2,3,4,5]

for i in myList:
    print(i)
```

Structs (C)

C Struct

Before classes (which combine attribute data and functionality), C would separate procedures (the functions) from data (variables).

But multiple variables of an entity (e.g. Student) can be defined in a ‘structure’ – otherwise known as the struct.

```
struct Student{
    char name[30];
    int id;
};
```

Classes (C++)

Classes in C++ vs Python

```cpp
class Student{
  private:
    int id;
    string name;

  public:
    Student(int id, string name){
        this.id = id;
        this.name = name;
    }
}
```

```python
class Student:
    def __init__(self, id, name):
        self.id = id
        self.name = name
```

Objects (C++)

Objects in C++ vs Python

```cpp
#include <stdio.h>
#include <Student.h>

int main() {
    Student nick(1234, "Nick");
    nick.print();
    return 0;
}
```

```python
from student import Student


nick = Student(1234, "Nick")
nick.print()
```

Objects in C++ vs Python

```cpp
#include <stdio.h>
#include <Student.h>

int main() {
    Student *nick = new Student(1234, "Nick");
    nick->print();
    return 0;
}
```

```python
from student import Student


nick = Student(1234, "Nick")
nick.print()
```

Inheritance

Inheritance in C++ vs Python

```cpp
class Child : public Parent{
  private:
    int id;
    string name;

  public:
    Student(int id, string name){
        this.id = id;
        this.name = name;
    }
}
```

```python
class Child(Parent):
    def __init__(self, id, name):
        self.id = id
        self.name = name
```

Summary

Summary

C was developed in the 1970s by Dennis Ritchie (evolved from B). It was procedural language so data was separated from procedures (functions).

C++ is the evolution (increment) of C, often known as ‘C with classes’, first released in 1985. The Object-Oriented Paradigm introduced the ability to encapsulate both variable data and functions (now methods) in one place.

C++ builds upon C so a lot of the original language features can be utilised (e.g. iostream) alongside all of the new OOP functionality.