ual: creative computing institute

# Week 3

C++

| | | |
|---|---|---|
| Day 1 | Tuesday 12-Sep-23 | 9am-11am + 6pm-7pm |
| Day 2 | Wednesday 13-Sep-23 | 9am-11am + 6pm-7pm |
| Day 3 | Thursday 14-Sep-23 | 9am-11am + 6pm-7pm |

Jasper Zheng

## About Jasper:

Hi, my name is Jasper Zheng

My works explore AI systems through their convergence with media and arts, primarily focusing on software systems that facilitate interactive and understandable human-AI co-creation. I am equally interested in the philosophical, ethical, and aesthetical implications inherent to the development of AI.

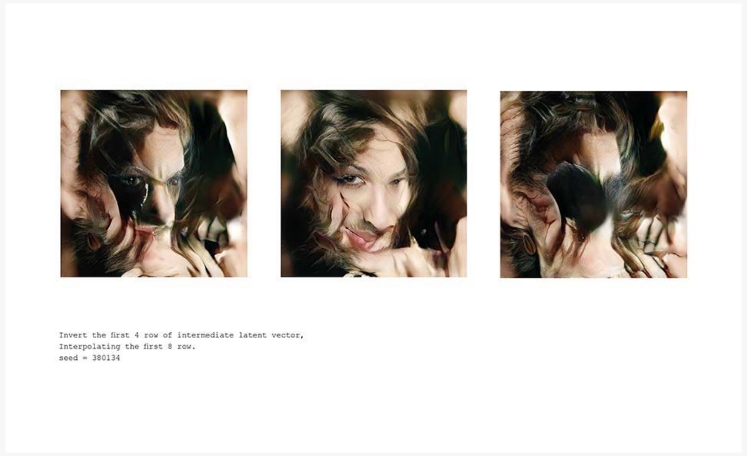
I am a PhD student in explainable AI and music technology at Queen Mary University of London. I studied at Creative Computing Institute in UAL (2021-22), Computer Science Department in University of Liverpool (2017-21).

https://jasperzheng.cc/

## The Third Dashboard (2019)

A data-reactive experiment tracks over one's YouTube account, produce moving image that reflects their personalised recommendation system.

https://jasperzheng.cc/works/dashboards



Invert the first 4 row of intermediate latent vector,
Interpolating the first 8 row.
seed = 380134

## Manipulated Network (2022)

Collection of quirks and oddities generated from corrupted generative models.

https://jasperzheng.cc/works/manipulated-network

## StyleGAN-Canvas (2023)

An augmented encoder for realtime deep generative models.

https://jasperzheng.cc/works/stylegan-canvas

ual: creative computing institute

Example of C++ Based Work

## What you will learn in these 3 days

- Basic building blocks in C++ (i.e. syntax / operations / functions...)
- A sense of coding - that is:
    - How to design and build a program?
    - How to map real-world behaviours/interactions into a program?
    - How to present a solution in a way that **both computer, and human can understand**.

## In the next 3 days...

Each day we will take a simple problem (e.g. a calculator, a X and O game) as example, design and implement a C++ program to solve them.

**1**

Setting up environment

Basic syntax / building blocks / inputs

Build a calculator

**2**

Array and loops

2D arrays and nested loops

Build a Tic-Tac-Toe game

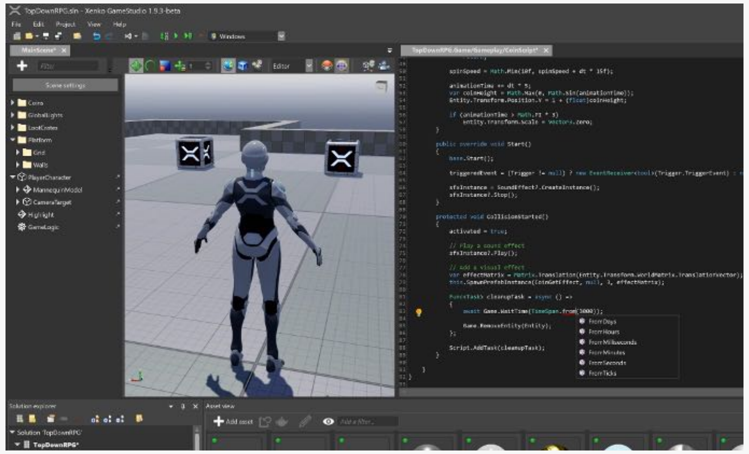
**3**

Object-oriented programming (a beautiful way to manage your program)

Build a better Tic-Tac-Toe game

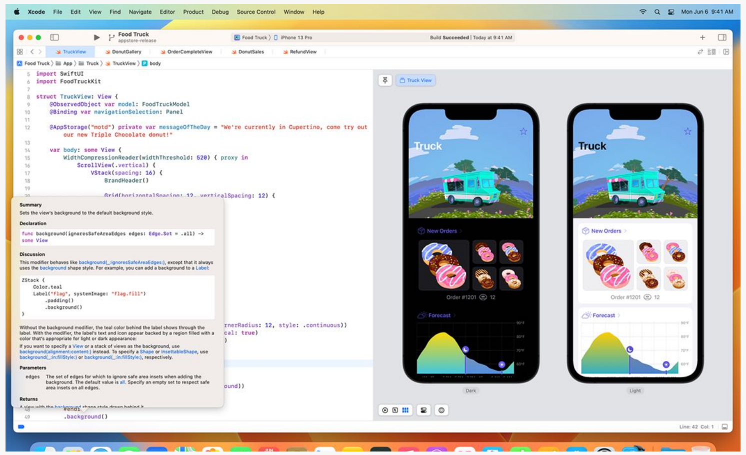
# How you will/may use C++ at CCI

C++ has been integrated into various development environments.
You're likely to see C++ code as part of your tools.



```
        spinSpeed = Math.Min(10f, spinSpeed + dt * 15f);

        animationTime += dt * 5;
        var coinHeight = Math.Max(0, Math.Sin(animationTime));
        Entity.Transform.Position.Y = 1 + (float)coinHeight;

        if (animationTime > Math.PI * 3)
            Entity.Transform.Scale = Vector3.Zero;
    }

    public override void Start()
    {
        base.Start();

        triggeredEvent = (Trigger != null) ? new EventReceiver<bool>(Trigger.TriggerEvent) :

        sfxInstance = SoundEffect?.CreateInstance();
        sfxInstance?.Stop();
    }

    protected void CollisionStarted()
    {
        activated = true;

        // Play a sound effect
        sfxInstance?.Play();

        // Add a visual effect
        var effectMatrix = Matrix.Translation(Entity.Transform.WorldMatrix.TranslationVector);
        this.SpawnPrefabInstance(CoinGetEffect, null, 3, effectMatrix);

        Func<Task> cleanupTask = async () =>
        {
            await Game.WaitTime(TimeSpan.from(1000));

            Game.RemoveEntity(Entity);
        };

        Script.AddTask(cleanupTask);
    }

  }
}
```

Use C++ to let your game characters move.


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Food Truck
appstore-release
Food Truck 〉 iPhone 13 Pro
TruckView
DonutGallery
OrderCompleteView
DonutSales
RefundView
Food Truck 〉 App 〉 Truck 〉 TruckView 〉 body
Truck View
Truck
New Orders
Forecast
Dark
Light

```
import SwiftUI
import FoodTruckKit

struct TruckView: View {
    @ObservedObject var model: FoodTruckModel
    @Binding var navigationSelection: Panel

    @AppStorage("motd") private var messageOfTheDay = "We're currently in Cupertino, come try out
        our new Triple Chocolate donut!"

    var body: some View {
        WidthCompressionReader(widthThreshold: 520) { proxy in
            ScrollView(.vertical) {
                VStack(spacing: 16) {
                    BrandHeader()

                    Grid(horizontalSpacing: 12, verticalSpacing: 12) {
```

**Summary**
Sets the view's background to the default background style.

**Declaration**

```
func background(ignoresSafeAreaEdges edges: Edge.Set = .all) ->
some View
```

**Discussion**
This modifier behaves like background(_:ignoresSafeAreaEdges:), except that it always uses the background shape style. For example, you can add a background to a Label:

```
ZStack {
    Color.teal
    Label("Flag", systemImage: "flag.fill")
        .padding()
        .background()
}
```

Without the background modifier, the teal color behind the label shows through the label. With the modifier, the label's text and icon appear backed by a region filled with a color that's appropriate for light or dark appearance:

If you want to specify a View or a stack of views as the background, use background(alignment:content:) instead. To specify a Shape or InsettableShape, use background(_:in:fillStyle:) or background(_:in:fillStyle:), respectively.

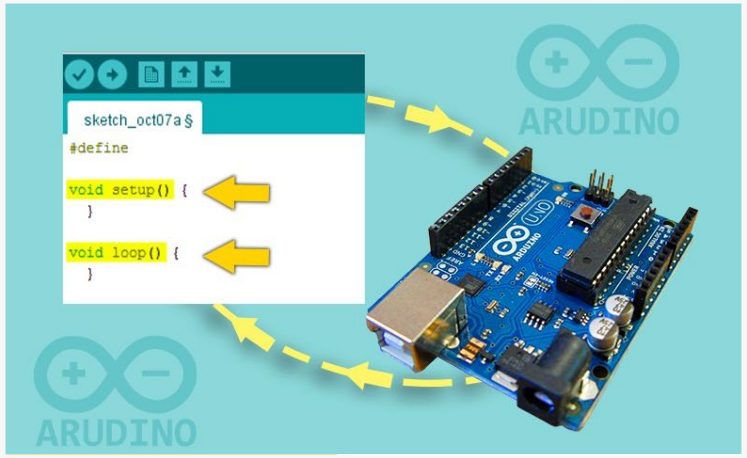
**Parameters**

edges The set of edges for which to ignore safe area insets when adding the background. The default value is all. Specify an empty set to respect safe area insets on all edges.

**Returns**

```
rnerRadius: 12, style: .continuous))
cal: true)
)
ound))
    #endif
    .background()
```

Use C++ to create interactions in your APP.



```
#define

void setup() {
}

void loop() {
}
```

Use C++ to control your physical circuit.

ual: creative computing institute

# The MODULAR Handbook

In which Units will you encounter C++?

- Creative Making: Advanced Physical Computing
- Coding Two: Advanced Frameworks
- Creative Making: Advanced Visualisation and Computational Environments

What are some examples of the work you will be expected to make?

- Immersive experiences (VR/AR)
- Games / interactive systems
- Mobile / desktop applications
- …

Link to Handbook: https://www.arts.ac.uk/__data/assets/pdf_file/0016/310723/MA_MSc-Computing-and-Creative-Industry-Modular-Book-of-Units-1.pdf

ual: creative computing institute

C++ Day 1

# Introduction and "Hello World"

2023

Jasper Zheng

"To better communicate to our computers **what exactly it is we want them to do**, we've developed C++ to make the communication process easier."

C++ official documentation

"To better communicate to our computers **what exactly it is we want them to do**, we've developed C++ to make the communication process easier."

C++ gives us low-level control over our machine (it's closer to the hardware you are running your code on).

| Pros | Cons |
|---|---|
| Known for performance<br><br>Useful for scalable applications<br><br>Useful for real-time audio and graphics (game development) | Complex syntax and file systems<br><br>Compiling a program can be tricky |

C++ gives us low-level control over our machine (it's closer to the hardware you are running your code on).

## What can you do with C++

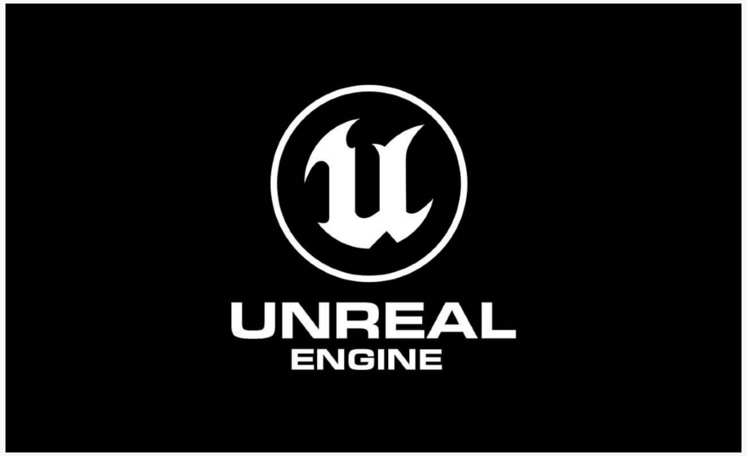
Usually, we are not using C++ as a standalone tool, instead we combine it with other tools as part of our projects.

Depending on your choices of platforms, software and hardware, C++ been integrated into different tools to create:

- software systems
- embedded physical systems
- playable digital experiences
- ...

### Unity (Game Engine)

Language: C# (C-based language very similar to C++)



### Unreal Engine (Game Engine)

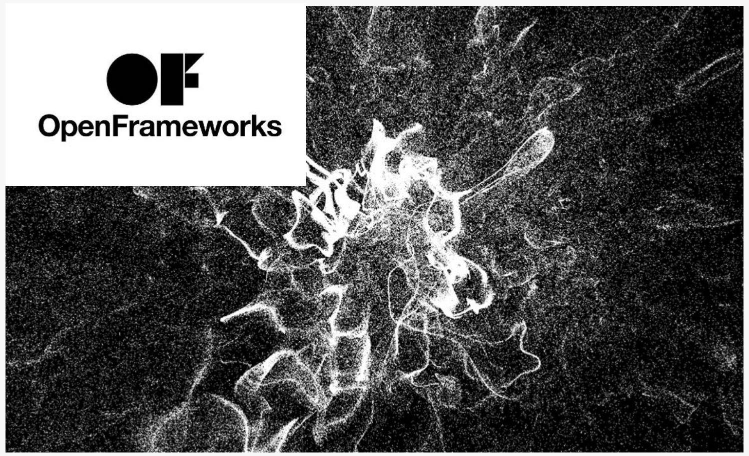
Language: C++

### Apple Platforms Dev (iOS, macOS, visionOS)

Language: Swift (Application programming language extended from C++)

JUCE (Audio/MIDI/Virtual Instruments)

Language: C++



OpenFrameworks (Creative Coding)

Language: C++

Arduino IDE (Physical Computing)

Language: C / C++ (A framework built on top of C++)

## What C++ is not ideal for

- Online, networked, web-based system (consider JavaScript?)
- Machine learning / data and statistics (consider Python to start with).

# Outlook Day 1

- Differences between C++ / JS / Python
- Hello World
- Console Interaction (Getting Inputs from users)
    - Example program: Console Calculator

## Before we start

There're some fundamental differences between C++ and Python/JavaScript...

Before we start...

## Compiler vs. Interpreter

### Compiled Language

A compiler takes entire program, converts it into machine code, then executed by the machine.

- Takes a compilation stage
- But usually run faster after compiled
- A single error would prevent the whole program from running.

C++ is a compiled language

### Interpreted Language

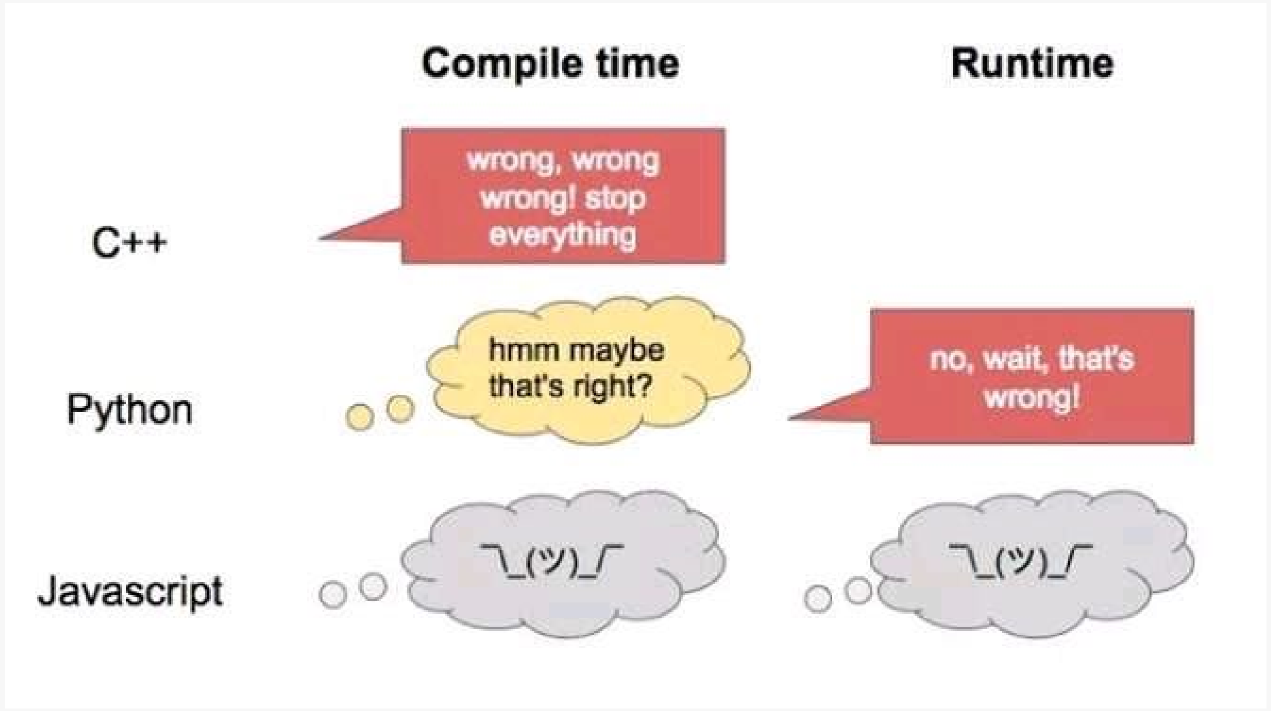
An interpreter directly executes instructions written in a script, line by line, without previously compiling.

- Easier to implement
- Can execute code "on the fly"
- Error occurred at run-time

JavaScript, Python are interpreted languages

Before we start...

## Compiler vs. Interpreter



Source https://medium.com/@simonravi/a-quick-look-into-javascript-python-go-and-c-6878c8ee0dde

Before we start...

# Statically-typed vs. Dynamically-typed

## Statically-typed Language

The type of a variable is known before compile time.

- We specify the data type of each variable
- And it's fixed after we declared them

```
int my_number = 10;
string my_number_as_text = "Ten";
```

C++ is a statically-typed language

## Dynamically-typed Language

The type of a variable is checked by the program.

- We don't need to declare the type of variables
- The type of a variable can be changed

```
my_number = 10
my_number = 'ten'
```

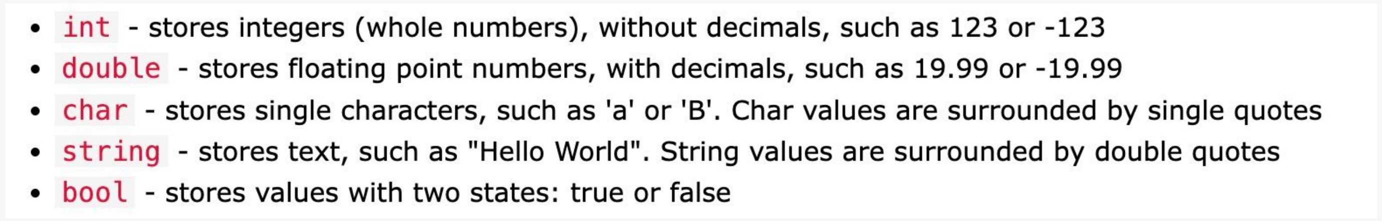
JavaScript, Python are dynamically-typed languages

Before we start...

## Recap: Data Types

You may have already seen different types of data in Python / JS (whole numbers, fractional numbers, text, true and false...),

Read the chapter on variables, see how to use them in C++:
https://www.w3schools.com/cpp/cpp_variables.asp



- `int` - stores integers (whole numbers), without decimals, such as 123 or -123
- `double` - stores floating point numbers, with decimals, such as 19.99 or -19.99
- `char` - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- `string` - stores text, such as "Hello World". String values are surrounded by double quotes
- `bool` - stores values with two states: true or false

Before we start...

# An Example C++ Program

## `#include <iostream>`

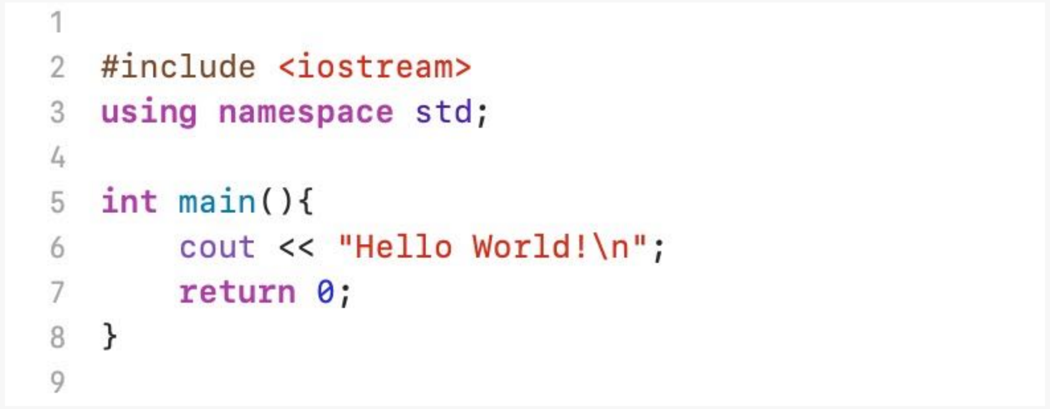
We use `#include` to tell the compiler which file/library to be included in the program. Here we are including the `iostream` library, which contains the `cout` functions that we are going to use in this program.

## `using namespace std;`

We are telling the compiler to use everything under the `std` (standard) namespace.

## `int main()`

Every C++ program must have a main function, the compiler will look for the `main()` function to execute your code. And `int` indicates that the function will return an integer.



```

#include <iostream>
using namespace std;

int main(){
    cout << "Hello World!\n";
    return 0;
}

```

# Daily Code Jumpstart Choreography

| | Mon | Tues | Wed | Thurs | Fri |
|---|---|---|---|---|---|
| 9am-10am | --- | Coaching aims | Daily Aims and Objectives | Daily Aims  and Objectives | --- |
| 10am-13oo | --- | Self-study Time | Self-study Time | Self-study Time | --- |
| Break | --- | Social Lunch | Social Lunch | Social Lunch | --- |
| 14oo-16oo | --- | Self-study Time | Self-study Time | Self-study Time | --- |
| 18oo-19oo | --- | QandA with Coach | QandA with Coach | QandA with Coach | --- |

## Day 1 Tasks

- Task 1 - Install and setup environment (approx. 45 mins)
- Task 2 - Get familiar with the syntax (approx. 10 mins)
- Task 3 - Getting user's input from console and build a calculator (approx. 60 mins)

Task 1

# 1.1 Install and Setup Environment

**Task 1.1 - Download and set up an IDE**

You will need an IDE (integrated development environment) to run and debug C++ applications.

An IDE is not just a text editor, it has the programming environment needed to compile a program.

If you're on MacOS: Xcode

If you're on Windows: Visual Studio, (make sure to select "**Desktop development with C++**" when installing)

If you're on Linux: You already have everything you need, just make sure your Linux distribution has the latest GCC.

Writing and Compiling C++ on Linux

Task 1

## 1.2 Create a "hello world" program

**Task 1.2 - Create a "hello world" program (approx. 15 mins)**

If you're on MacOS, follow this tutorial:

Set up a program in Xcode

If you're on Windows, follow this tutorial

Set up a program in Visual Studio.

Task 2

## 2. Get Familiar with the Syntax

**Task 2 - Basic Building Blocks (approx. 10 mins)**

You might already be familiar with **data types** (int, float...) and **operators** (+, -, &&, ==…) in JS and Python.

Those concepts are the same in C++, with a few differences in the syntax.

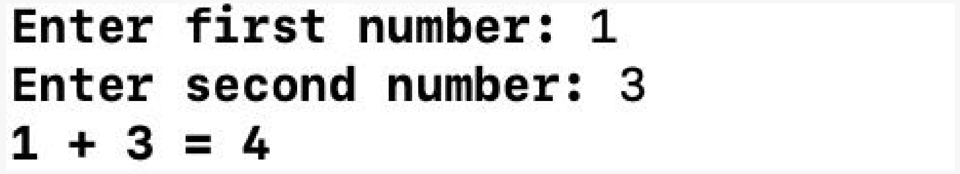
Therefore, in this task, have a quick look at [Chapter 2 Using Data in C++]

Task 3

# 3. Getting User's Input from Console

**Task 3.1 - Build a Basic Calculator (approx. 25 mins)**

In this task, we are going to build a console calculator that asks the user to enter two numbers and output the sum of them. An ideal program is shown in the screenshot ->



```
Enter first number: 1
Enter second number: 3
1 + 3 = 4
```

Take this as a chance to get familiar with the syntax and the routine of compiling and running a C++ program.

You can try make it on your own, or follow these video tutorial:

**Step 1 - Getting User's Input (approx. 15 mins)**

Watch and follow this tutorial from 59:41 to 1:09:31

**Step 2 - Build a Calculator (approx. 10 mins)**

Watch and follow this tutorial from 1:55:58 to 2:02:20

Task 3

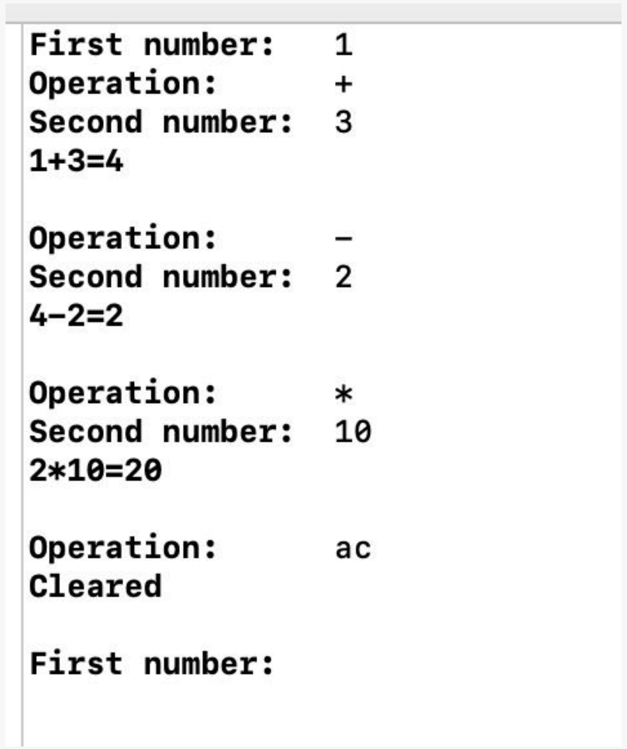
# 3. Getting User's Input from Console

**Task 3.2 - Build a Better Calculator**

- Can we extend the calculator to more operators? (e.g., +, -, /, *)
- Can we continuously take inputs from users and apply operations on the previous result?
- [optional] Can we add an "AC" (all clear) command like a regular calculator, which clear the previous result and start a new session when the user types "ac" as an operator.
- The ideal system is shown in the screenshot ->

Hint: a `while (true)` loop might be a good choice, also think about the flowchart of your program

While Loop

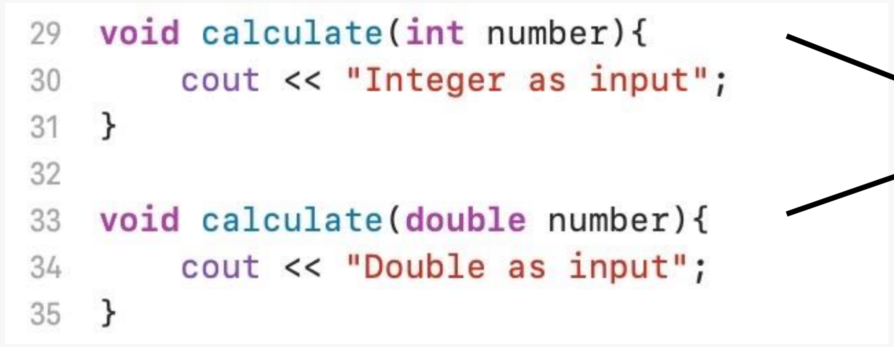


```
First number:    1
Operation:       +
Second number:   3
1+3=4

Operation:       -
Second number:   2
4-2=2

Operation:       *
Second number:   10
2*10=20

Operation:       ac
Cleared

First number:
```

# Common Questions

**Functions in C++**

Parameter types really matters



```cpp
void calculate(int number){
    cout << "Integer as input";
}

void calculate(double number){
    cout << "Double as input";
}
```

These are different functions

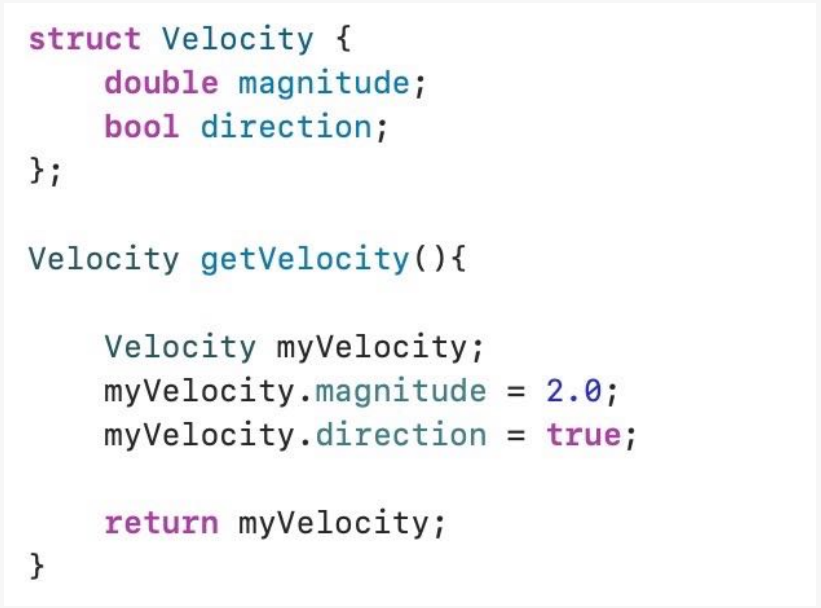
Actually it’s called function polymorphism (A function behaves differently in different situations).

Further reading: [Polymorphism in C++]()

# Common Questions

## Return Multiple Variables

`struct` is what you'll need. Structures are a way to group several related variables into one place.

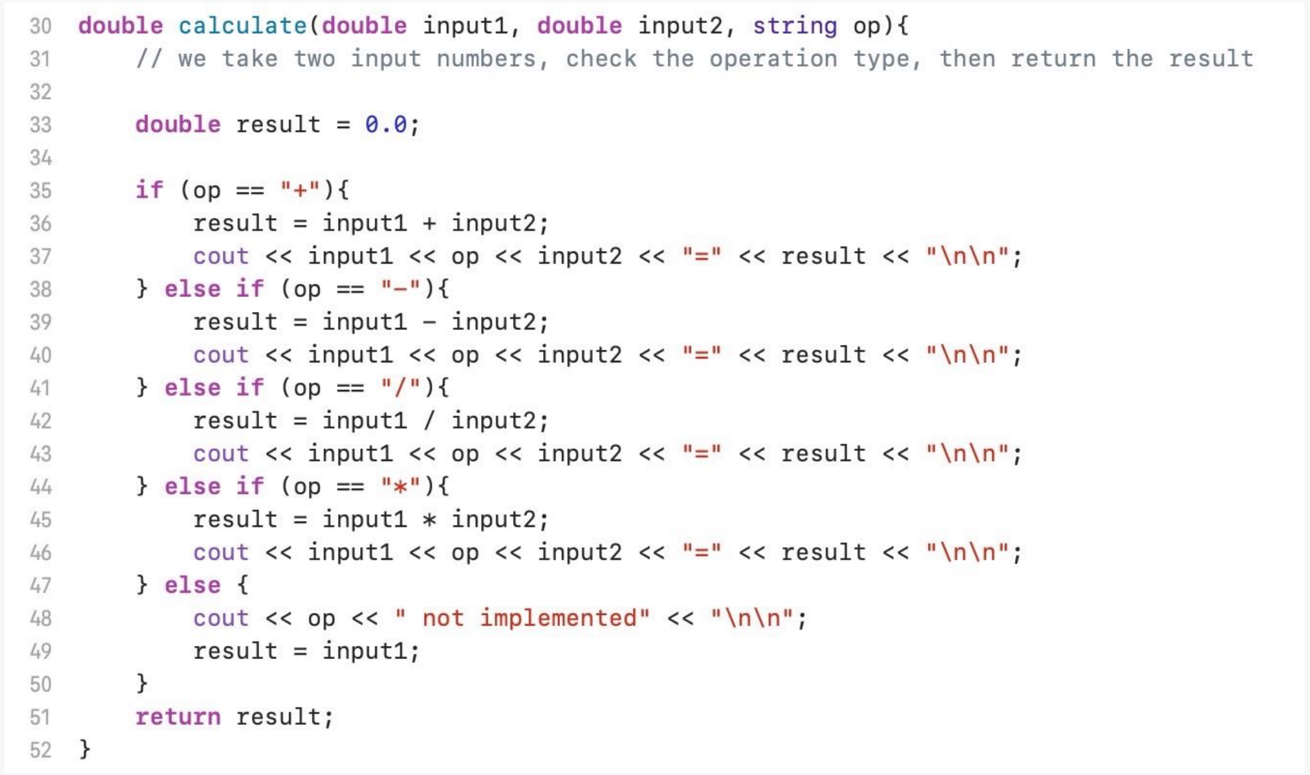


```cpp
struct Velocity {
    double magnitude;
    bool direction;
};

Velocity getVelocity(){

    Velocity myVelocity;
    myVelocity.magnitude = 2.0;
    myVelocity.direction = true;

    return myVelocity;
}
```

Solution

## Task 3.2 – Solution

### Step 1 - Extend the calculator to more operators



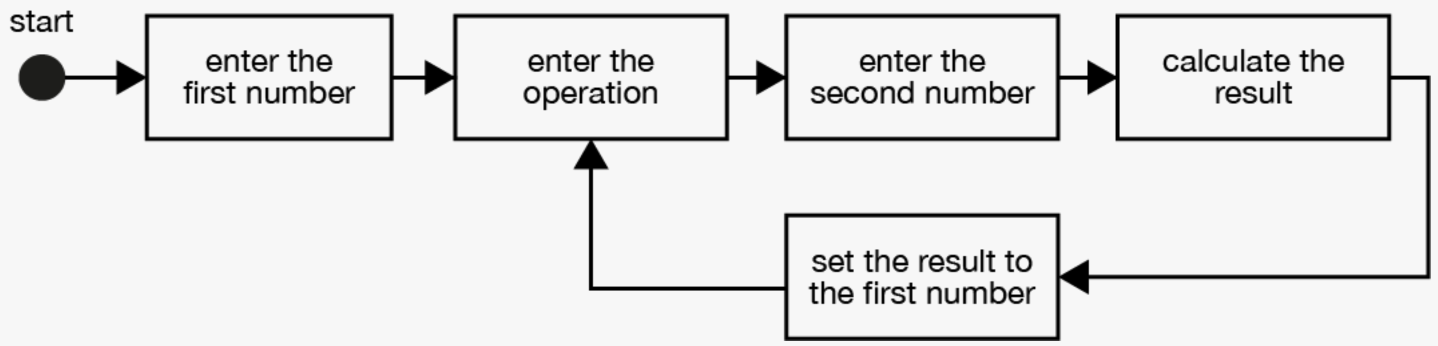
```cpp
double calculate(double input1, double input2, string op){
    // we take two input numbers, check the operation type, then return the result

    double result = 0.0;

    if (op == "+"){
        result = input1 + input2;
        cout << input1 << op << input2 << "=" << result << "\n\n";
    } else if (op == "-"){
        result = input1 - input2;
        cout << input1 << op << input2 << "=" << result << "\n\n";
    } else if (op == "/"){
        result = input1 / input2;
        cout << input1 << op << input2 << "=" << result << "\n\n";
    } else if (op == "*"){
        result = input1 * input2;
        cout << input1 << op << input2 << "=" << result << "\n\n";
    } else {
        cout << op << " not implemented" << "\n\n";
        result = input1;
    }
    return result;
}
```

Solution

## Task 3.2 – Solution

**Step 2 - Continuously taking user's input**

First, let’s draw a flowchart of our program.

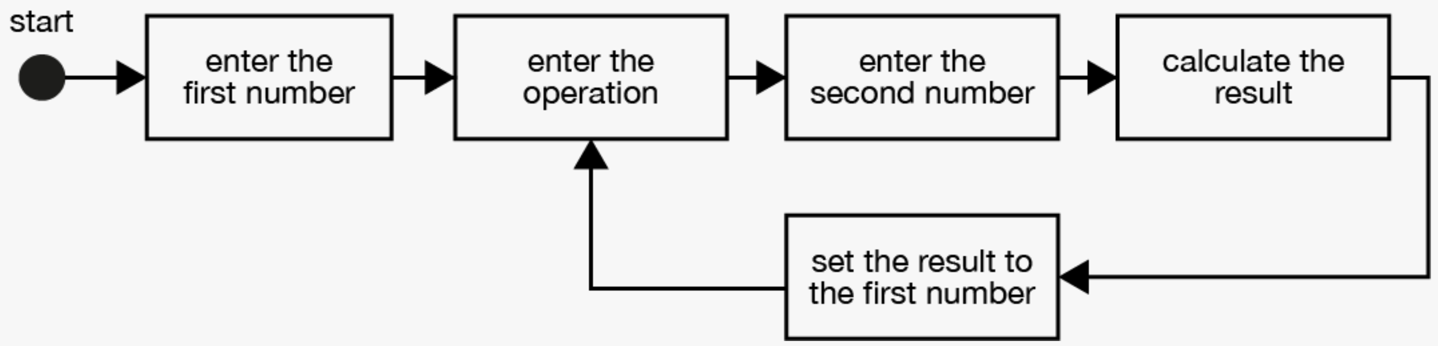


Solution

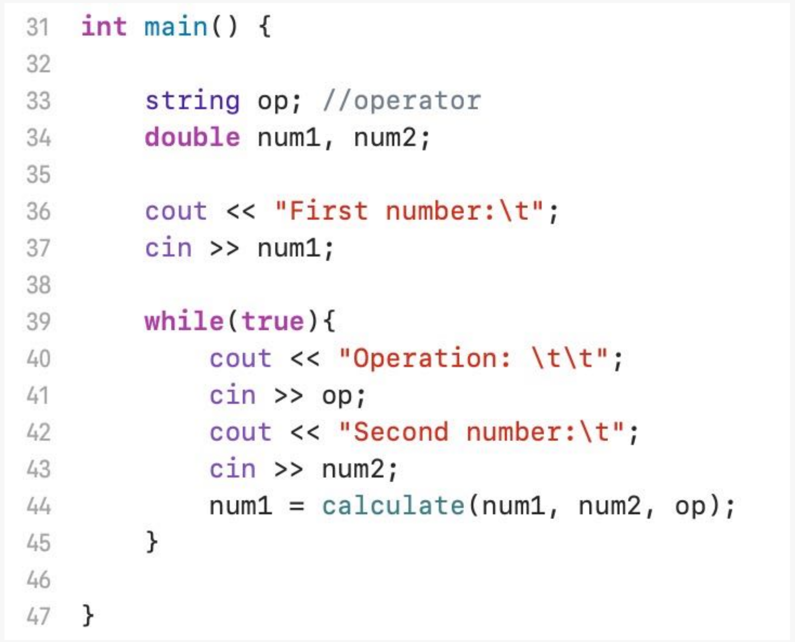
# Task 3.2 – Solution

### Step 2 - Continuously taking user's input

First, let's draw a flowchart of our program.



Then, translate our flowchart into codes:



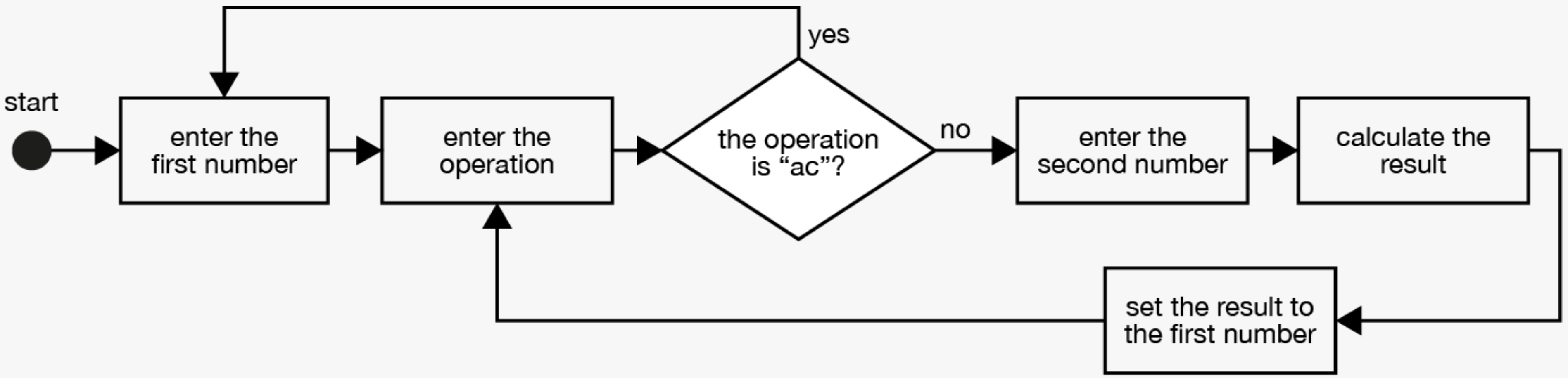
```cpp
int main() {

    string op; //operator
    double num1, num2;

    cout << "First number:\t";
    cin >> num1;

    while(true){
        cout << "Operation: \t\t";
        cin >> op;
        cout << "Second number:\t";
        cin >> num2;
        num1 = calculate(num1, num2, op);
    }

}
```

Solution

# Task 3.2 – Solution

**Step 3 - The "AC" command**

Let’s extend our flowchart.



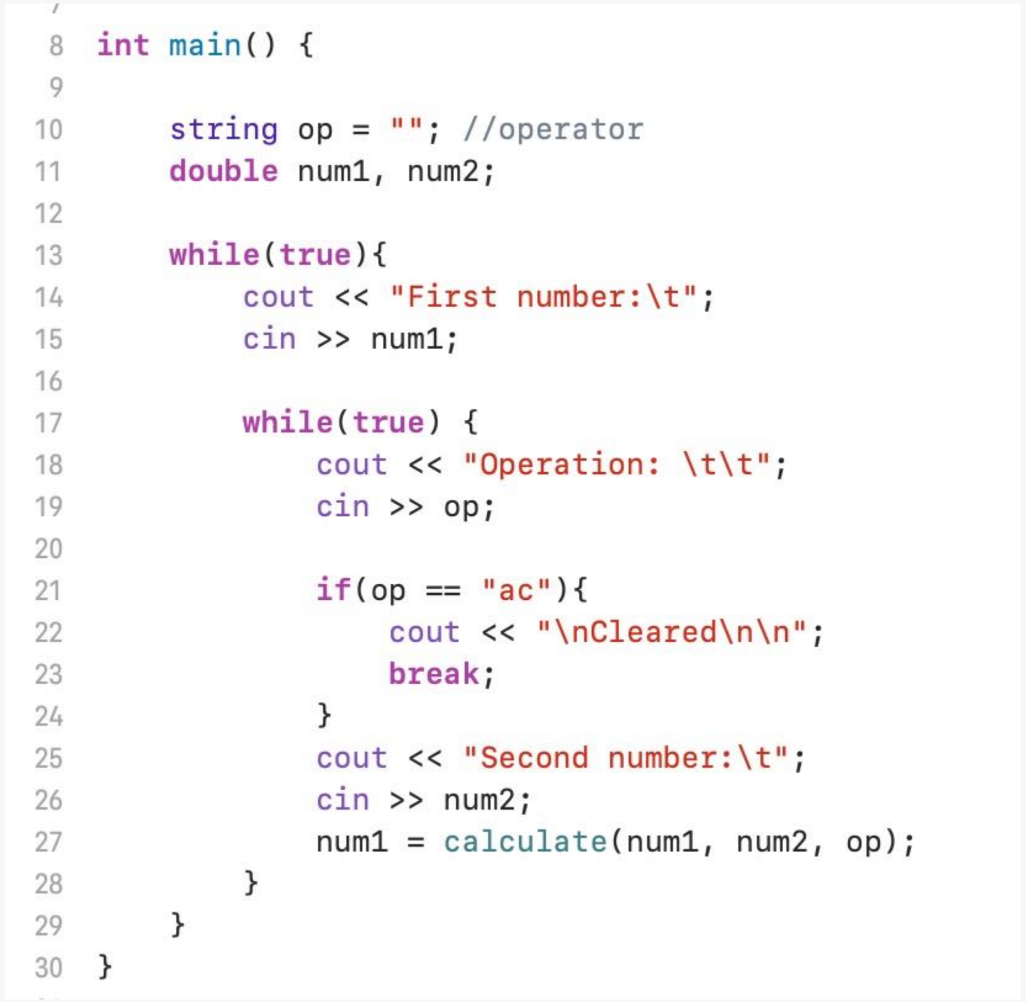
Solution

# Task 3.2 – Solution

**Step 3 - The "AC" command**

Translate the flowchart into codes.

Full code



```
int main() {

    string op = ""; //operator
    double num1, num2;

    while(true){
        cout << "First number:\t";
        cin >> num1;

        while(true) {
            cout << "Operation: \t\t";
            cin >> op;

            if(op == "ac"){
                cout << "\nCleared\n\n";
                break;
            }
            cout << "Second number:\t";
            cin >> num2;
            num1 = calculate(num1, num2, op);
        }
    }
}
```

# Day 1 Resources

C++ documentation https://www.cplusplus.com

C++ tutorials on W3School https://www.w3schools.com/cpp/default.asp

★★★ C++ Cheat Sheet ★★★

https://github.com/mortennobel/cpp-cheatsheet

Further reading: compiler and interpreter (optional)
https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/

Further reading: Statically v. dynamically v. strongly v. weakly typed languages (optional)
https://www.educative.io/answers/statically-v-dynamically-v-strongly-v-weakly-typed-languages

## Day 1 Deliverables

Have an IDE set up on your machine and make sure your Hello World program runs

Familiar with the routine of creating a C++ program

Familiar with the basic syntax in C++

## Day 1 de-brief

- How was today for you?
- What has gone well?
- What went as planned?
- What surprised you?
- Did you find today difficult?

- Share anything you made?
-

  Ask around the class and see if they have anything to share?

## Outlook day 2

Tomorrow we'll first look at example solutions to our calculator (Task 3.2), and also share some of your work if you would like to.

Then we'll be looking at arrays and loops. We'll also start building a simple Tic-Tac-Toe game.

See you tomorrow!

ual: creative computing institute

## Day 1 Survey

https://artslondon.padlet.org/hbrueggemann/j2yr3zfwkap4v4rq

The password is **Jumpstart**.

Thank you, see you tomorrow at TIME