# C++ Built-in methods

In this lesson, we'll discuss some essential built-in C++ methods.

**We'll cover the following**



## Min / Max #

Compare two integers and get minimum or maximum using a built-in function.

```
int a = 7;
int b = 4;
int M = max(a, b) // 7
int m = min(a, b) // 4
```

## Swap #

Use the built-in `swap` method to swap two variables of the same type.

```
int a = 5, b = 8;
swap(a, b);
```

## Auto #

The `auto` keyword automatically derives the data type from the initialization so you don’t have to.

This is very handy at times when the data type is complex, like a vector of pair of integers

```
vector<pair<pair<int,int>,pair<int,int>>> v;
```

To iterate using a `for` loop we would do this:

```
for (pair<pair<int,int>,pair<int,int>> it : v)
```

Which can make the code unreadable and slow you down. The same happens when using the `auto` keyword.

```
for (auto it : v)
```

---

# Pair #

Many times, `pair` is a handy alternative to defining `struct` or `class`.

Suppose you need a structure to define a point, meaning you need an `x` and a `y` coordinate.

Using a structure:

```
struct Point {
    int x;
    int y;
}
```

This can be simplified using `pair`.

There are multiple ways to initialize.

```
pair<int,int> coord = {x, y}
pair<int,int> coord = make_pair(x, y)
```

Access members using the `first` and `second` keyword:

```
pair<int,int> coord = {x, y}
int x = coord.first
int y = coord.second
```

You can also use different data types for the first and second member and compose a pair inside a pair. For example:

---

```
pair<string, pair<int, int> > var = {"abc",{4, 5} }
```

In the next lesson, we'll discuss how input and output works in C++.