# C++ Utilities

https://en.cppreference.com/w/cpp/utility

C++ includes a variety utility libraries that provide functionality ranging from bit counting to partial function implementation.
These libraries can be broadly divided into two groups:

- Language support libraries
- General purpose libraries

## Language Support

Provide classes and functions that interact closely with language features and support common language idioms

- Type support (std::size_t)
- Dynamic memory management (std::shared_ptr)
- Error Handing (std::exception, assert)
- Initializer list (std::vector{1, 2})

## General Purpose Utilities

- Program utilities (std::abort)
- Date and Time (std::chrono::duration)
- Optional, Variant (std::variant)
- Pairs and tuples (std::tuple)
- Swap, forward and move (std::move)

## std::swap

```cpp
int main() {
  int a = 3;
  int b = 5;

  // before
  std::cout << a << ' ' << b << '\n';

  std::swap(a, b);

  // after
  std::cout << a << ' ' << b << '\n';
}
```

### Output:

```
3 5
5 3
```

## std::variant

```cpp
int main() {
  std::variant<int, float> v1;
  v1 = 12;  // v contains int
  cout << std::get<int>(v1) << endl;
  std::variant<int, float> v2{3.14F};
  cout << std::get<1>(v2) << endl;

  v2 = std::get<int>(v1);  // assigns v1 to v2
  v2 = std::get<0>(v1);    // same as previous line
  v2 = v1;                 // same as previous line
  cout << std::get<int>(v2) << endl;
}
```

### Output:

```
12
3.14
12
```

## std::any

```cpp
int main() {
  std::any a;  // any type

  a = 1;  // int
  cout << any_cast<int>(a) << endl;

  a = 3.14;  // double
  cout << any_cast<double>(a) << endl;

  a = true;  // bool
  cout << std::boolalpha << any_cast<bool>(a) << endl;
}
```

### Output:

```
1
3.14
true
```

## std::optional

```cpp
std::optional<std::string> StringFactory(bool create) {
  if (create) {
    return "Modern C++ is Awesome";
  }
  return {};
}

int main() {
  cout << StringFactory(true).value() << '\n';
  cout << StringFactory(false).value_or(":(") << '\n';
}
```

### Output:

```
Modern C++ is Awesome
:(
```

## std::tuple

```cpp
int main() {
  std::tuple<double, char, string> student1;
  using Student = std::tuple<double, char, string>;
  Student student2{1.4, 'A', "Jose"};
  PrintStudent(student2);
  cout << std::get<string>(student2) << endl;
  cout << std::get<2>(student2) << endl;

  // C++17 structured binding:
  auto [gpa, grade, name] = make_tuple(4.4, 'B', "");
}
```

### Output:

```
GPA: 1.4, grade: A, name: Jose
Jose
Jose
```

## std::chrono

```cpp
#include <chrono>

int main() {
  auto start = std::chrono::steady_clock::now();
  cout << "f(42) = " << fibonacci(42) << '\n';
  auto end = chrono::steady_clock::now();

  chrono::duration<double> sec = end - start;
  cout << "elapsed time: " << sec.count() << "s\n";
}
```

### Output:

```
f(42) = 267914296
elapsed time: 1.84088s
```