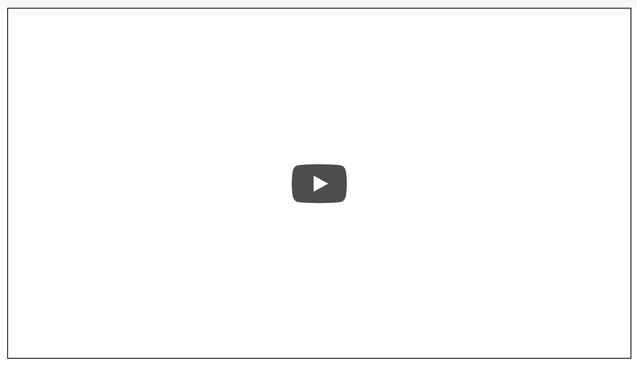


https://youtu.be/xBO3kdZTHyc

When passing parameters to a function in C++, there is a variety of strategies a programmer can choose from. In this section, we will take a look at these in turn from the perspective of stack usage. First, we will briefly revisit the definition of scope, as well as the strategies call-by-value and call-by-reference. Then, we will look at the amount of stack memory used by these methods.

> Note: Click "Expand" at the bottom left of the workspace below for better readability and overall workspace experience.

Guide

Variable Scopes in C++

The time between allocation and deallocation is called the **lifetime** of a variable. Using a variable after its lifetime has ended is a common programming error, against which most modern languages try to protect: Local variables are only available within their respective scope (e.g. inside a function) and are simply not available outside - so using them inappropriately will result in a compile-time error. When using pointer variables however, programmers must make

Page 1 of 10

scope.cpp

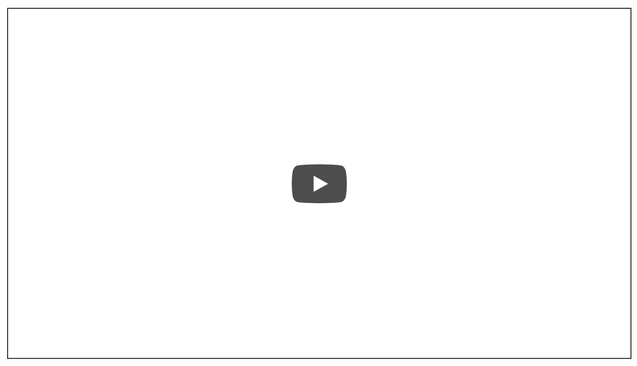
```
1  bool MyLocalFunction(int myInt)
2  {
3      bool isBelowThreshold = myInt < 42 ? true : false;
4      return isBelowThreshold;
5  }
6
7  int main()
8  {
9      bool res = MyLocalFunction(23);
10     return 0;
11 }
```

root@8cdf3aa2e41f: /home/w

```
root@8cdf3aa2e41f:/home/workspace#
```

↑ Menu Expand

Outro



https://youtu.be/S4NSWgZyvT4