

# Const-Guideline

Before you declare a variable, think about whether its value will be changed later or not!

If not, use the keyword `const` to declare the variable as constant.

# const – Necessary?

- Protects against unintended changes

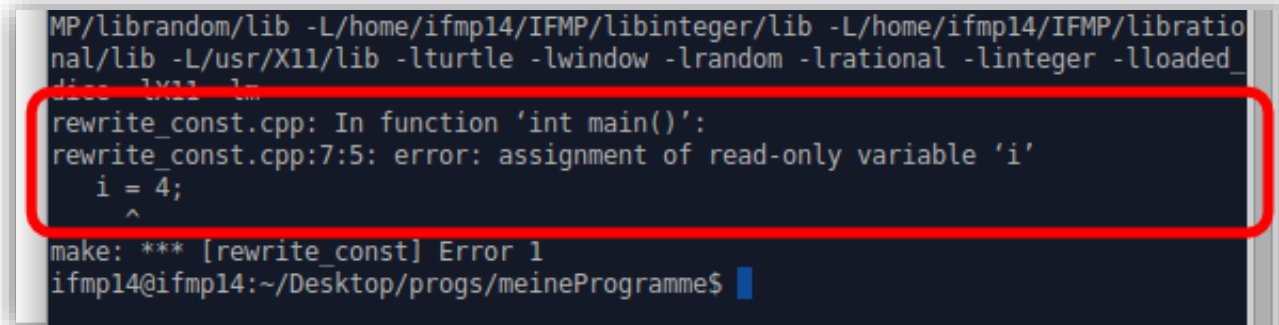
# const – Necessary?

- Protects against unintended changes
  - Compiler error message

```
MP/librandom/lib -L/home/ifmp14/IFMP/libinteger/lib -L/home/ifmp14/IFMP/libratio
nal/lib -L/usr/X11/lib -lturtle -lwindow -lrandom -lrational -linteger -lloaded_
dice -lX11 -lm
rewrite_const.cpp: In function 'int main()':
rewrite_const.cpp:7:5: error: assignment of read-only variable 'i'
    i = 4;
    ^
make: *** [rewrite_const] Error 1
ifmp14@ifmp14:~/Desktop/progs/meineProgramme$
```

# const – Necessary?

- Protects against unintended changes
  - Compiler error message

A terminal window showing the compilation of a C++ program. The command line includes various library flags. The output shows a compiler error in the file 'rewrite\_const.cpp' at line 7, column 5. The error message is 'error: assignment of read-only variable 'i''. The code snippet shows 'i = 4;' with a caret under the 'i'. The terminal prompt is 'ifmp14@ifmp14:~/Desktop/progs/meineProgramme\$'.

```
MP/librandom/lib -L/home/ifmp14/IFMP/libinteger/lib -L/home/ifmp14/IFMP/libratio
nal/lib -L/usr/X11/lib -lturtle -lwindow -lrandom -lrational -linteger -lloaded_
dice -lX11 -lm
rewrite_const.cpp: In function 'int main()':
rewrite_const.cpp:7:5: error: assignment of read-only variable 'i'
    i = 4;
    ^
make: *** [rewrite_const] Error 1
ifmp14@ifmp14:~/Desktop/progs/meineProgramme$
```

- Communicate to reader
  - Reader knows: value will not change

# Example

Make this `const`-correct.

## 1. Program:

```
#include <iostream>
int main ()
{
    const int a = 5;
    std::cin >> a;
    std::cout << a + 5;

    return 0;
}
```

# Example

## Problem:

input operator `>>` changes **constant** variable

## 1. Program:

```
#include <iostream>
int main ()
{
    const int a = 5;
    std::cin >> a;
    std::cout << a + 5;

    return 0;
}
```



## Solution:

```
#include <iostream>
int main ()
{
    int a = 5;
    std::cin >> a;
    std::cout << a + 5;

    return 0;
}
```

# Example

Make this `const`-correct.

## 2. Program:

```
int main ()
{
    const int a = 5;
    int b = 2*a;
    int c = 2*b;
    b = b*b;

    return 0;
}
```

# Example

## Problem:

- `c` should be `const`.
- `c` is initialized without a later use.

## 2. Program:

```
int main ()
{
    const int a = 5;
    int b = 2*a;
    int c = 2*b;
    b = b*b;

    return 0;
}
```



## Solution:

```
int main ()
{
    const int a = 5;
    int b = 2*a;
    const int c = 2*b;
    b = b*b;

    return 0;
}
```



# Example

Make this `const`-correct.

## 3. Program:

```
int main ()
{
    const int a = 5;
    a = 5;

    return 0;
}
```

# Example

## Problem:

`a = 5;` overwrites `a` with **same** value.  
But `a` is `const`; `const` prevails.

## 3. Program:

```
int main ()
{
    const int a = 5;
    a = 5;

    return 0;
}
```



## Solution:

Remove `const` or  
`a = 5;`