Semester HS2020

C++ Zusammenfassung

Joel Schaltegger

24. Dezember 2020

Inhaltsverzeichnis

1 Introduction to C++

In C++ gibt es keinen Garbage Collector, wie man es aus anderen Sprachen, wie Java oder C# kennt. Warnung: Wenn Code "falsch"geschrieben wurde, kann Undefined Behavior auftreten.

1.1 C++ Compilation Process

*.cpp Files

- Also called Implementation File
- For function implementations (can be in .h as well)
- Source of compilation

*.h File

- Also called Header File
- Declarations and definitions to be used in other implementation files

3 Phases of compilation

- Preprocessor Textual replacement of preprocessor directives (include)
- Compiler Translation of C++ code into machine code (source file to object file)
- Linker Combination of object files and libraries into libraries and executables

1.2 Declarations and Definitions

All things with a name that you use in a C++ program must be declared before you can do so.

One Definition Rule

While a program element can be declared several times without problem there can be only one definition of it. This is called the **One Definition Rule** (ODR)!

Include Guard

Include guards ensure that a header file is only included once. Multiple inclusions could violate the One Definition Rule when the header contains definitions.

```
#ifndef SAYHELLO_H_
#define SAYHELLO_H_

#include <iosfwd>
struct Greeter { /* Some Code */ };

#endif /* SAYHELLO_H_ */
```

2 Values and Streams

2.1 Variable Definitions

- Defining a variable consists of specifying its type, its variable name and its initial value. E.g. int x{42};
- Empty braces mean default initialization. E.g. double x{};
- Using = for initialization we can have the compiler determine its type. E.g. auto const i = 5;

Constants

- Adding the const keyword in front of the name makes the variable a single assignment variable, aka a constant. E.g. int const x{42};
  - Must be initialized and immutable
- Use the keyword constexpr if the variable is required to be fixed at compile time. E.g. double constexpr pi{3.14159};

Why shoud I use const?

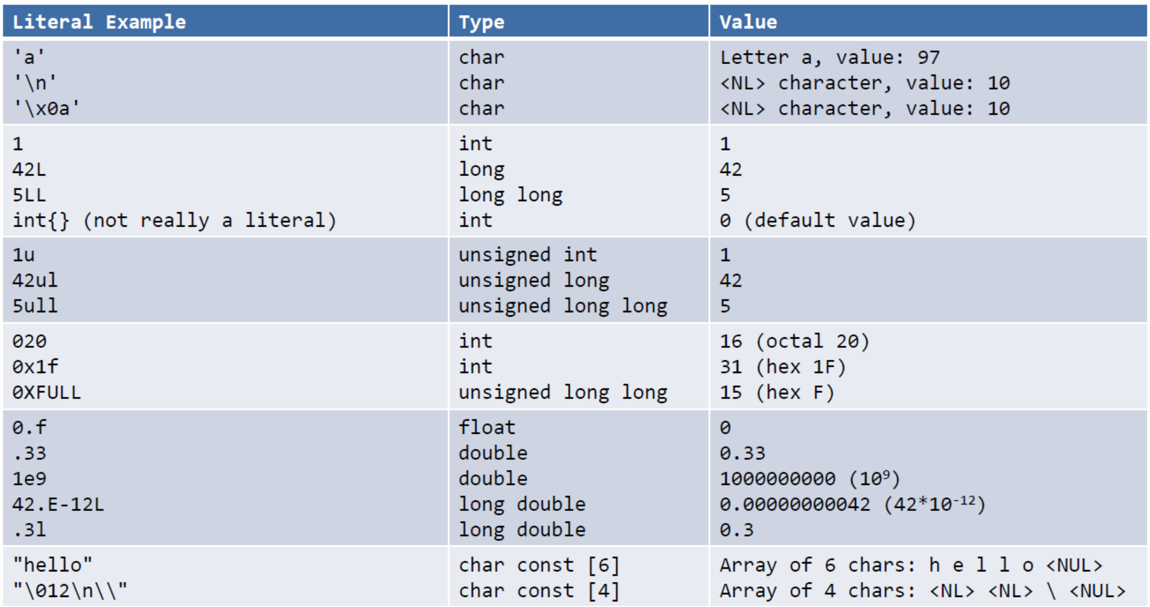
- A lot of code needs names for values, but often does not intend to change it
- It helps to avoid reusing the same variable for different purposes (code smell)
- It creates safer code, because a const variable cannot be inadvertently changed
- It makes reasoning about code easier
- Constness is checked by the compiler
- It improves optimization and parallelization (shared mutable state is dangerous)

Important types for Variable

- short, int, long, long long - each also available as unsigned version
- bool, char, unsigned char, signed char
- float, double, long double
- void is special, it is the type with no values
- class defined: E.g. std::string, std::vector

2.2 Values and Expressions

Integer to boolean: 0 = False, every other value = True
if (a < b < c) → zuerst wird a < b ausgewertet (true oder false). Dann wird der Boolean mit einem int (c) verglichen. Der Bool wird dafür implizit in 0 oder 1 gecastet.



Literal Example	Type	Value
'a' '\n' '\x0a'	char char char	Letter a, value: 97 <NL> character, value: 10 <NL> character, value: 10
1 42L 5LL int{} (not really a literal)	int long long long int	1 42 5 0 (default value)
1u 42ul 5ull	unsigned int unsigned long unsigned long long	1 42 5
020 0x1f 0XFULL	int int unsigned long long	16 (octal 20) 31 (hex 1F) 15 (hex F)
0.f .33 1e9 42.E-12L .3l	float double double long double long double	0 0.33 1000000000 (10^9) 0.00000000042 ($42*10^{-12}$) 0.3
"hello" "\012\n\\"	char const [6] char const [4]	Array of 6 chars: h e l l o <NUL> Array of 4 chars: <NL> <NL> \ <NUL>

2.3 Strings and Sequences

std::string is C++'s type for representing sequences of char (which is often only 8 bit) and are mutable. That means, we can modify the content. (Vergleich zu Java: Dort würde ein neues String Objekt erstellt werden)
Grundsätzlich werden Strings also als char const[ ] abgespeichert. Mit dem namespace std::literals hat man die Option hinter dem String eine 's' anzufügen, um das Objekt effektiv als String zu speichern. z.B. "ab"s

toUpper Iterator

```cpp
void toUpper(std::string & value) {
    transform(cbegin(value), cend(value), begin(value), ::toupper);
}
```

2.4 Input and Output Streams

Functions taking a stream object must take it as a reference, because they provide a side effect to the stream (i.e., output characters).

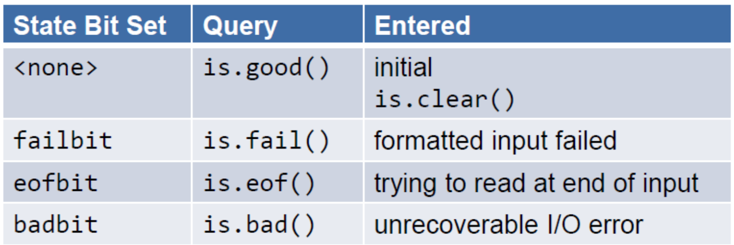
Reading from Input

- Reading into a std::string always works. Unless the stream is already !good() → Spaces werden übersprungen (neues String-Objekt)!
- Reading into other types (e.g. int) has no error recovery. A wrong input puts the stream into status fail and the characters remain in the input.
- Post-read check: if (in » age) { ... }
- Multiple subsequent reads are possible: if (in » symbol » count) { ... }
- Remove fail flag: in.clear()
- Ignore one char: in.ignore();
- Helpfull for reading: while (in.good()) um die Leseoperationen setzen.

Robust reading of an int value

```cpp
// Use an std::istringstream as intermediate stream
int inputAge(std::istream & in)
    std::string line{}
    while (getline(in, line)) {
        std::istringstream is{line};
        int age{-1};
        if (is >> age) {
            return age;
        }
    }
    return -1;
}
```

Stream States



State Bit Set	Query	Entered
<none>	is.good()	initial is.clear()
failbit	is.fail()	formatted input failed
eofbit	is.eof()	trying to read at end of input
badbit	is.bad()	unrecoverable I/O error

3 Woche03

4 Woche04

5 Woche05

6 Woche06

7 Woche07

8 Woche08

9 Woche09

10 Woche10

11 Woche11

12 Woche12

13 Woche13

14 Anhang

14.1 Übungen Woche XX

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Includes

```cpp
// Only the declaration for input and output streams
#include <iosfwd>

// Implementation of input stream
#include <istream>

// Implementation of output stream
#include <ostream>

// Declaration of both streams and additionally std::cout, std::cin, std::cerr
#include <iostream>

// Functions: std::tolower(c), std::isupper(c)
#include <cctype>

// Strings
#include <string>
```