

0. Einführung & Motivation

Ansatz: "C++ für Java-Kenner"

- Konzentration auf semantische Unterschiede 'gleichartiger' Konzepte
- Erörterung der C++ -spezifischen Konzepte (Overloading, Templates)

Anspruch auf Vollständigkeit

Sprache laut Standard **ISO/IEC 14882 von 2011**
incl. Standardbibliothek (STL and more)

Schwerpunkt auf Diskussion von Konzepten anhand von Beispielen

0. Einführung & Motivation

Warum noch eine OO-Sprache?

- die zudem
 - syntaktisch sehr ähnlich zu Java ist
 - älter ist als Java ...
 - 'gefährlicher' ist als Java ...

Weil C++ eine Sprache ist

- die
 - syntaktisch sehr ähnlich zu Java ist
 - älter ist als Java ...
 - 'gefährlicher' ist als Java ...

-- und potenziell effizienteren Code ermöglicht

<http://www.research.att.com/~bs/applications.html>

0. Einführung & Motivation

Java

- zahlreiche Sicherheitsvorkehrungen
kosten Zeit & Raum
- virtual machine
- architekturneutral



C++

- keinerlei Sicherheitsvorkehrungen
Reserven für Zeit & Raum
- native code
- architekturabhängig



Ein erster Blick: Hello World

Java

Hello.java

```
class Hello {  
    public static void main(String s[]) {  
        if (s.length < 1) return;  
        Hello h = new Hello(", " + s[0]);  
        h.speak();  
    }  
    String what;  
    void speak() {  
        System.out.println("Hello" + what);  
    }  
    Hello(String s) {  
        this.what = s;  
    }  
    protected void finalize() {  
        System.out.println("bye, bye");  
    }  
}
```

Ein erster Blick: Hello World

```
#include <iostream>
#include <string>

class Hello {
public: static void main(int c, char* v[]) {
        if (c < 2) return;
        Hello h = Hello(", "+std::string(v[1]));
        h.speak();
    }

private: std::string what;
        virtual void speak() {
            std::cout << "Hello" + what << std::endl;
        }
        Hello(std::string s) {
            this->what = s;
        }
        ~Hello() {
            std::cout << "bye, bye" << std::endl;
        }
}; // !!!!!!!!!!!!!
```

C++
h.cc

Java-Style

Ein erster Blick: Hello World

C++
h.cc

Java-Style

```
// ... continued  
  
int main(int argc, char* argv[])  
{  
    Hello::main(argc, argv);  
}
```

Ein erster Blick: Hello World

C++
h0.cc

C - Style

```
#include <cstdio>

int main(int argc, char* argv[])
{
    if (argc > 1)
        std::printf("Hello, %s\n", argv[1]);
}
```

Ein erster Blick: Hello World

- ☞ in C++ kann man offenbar Java-like (OO) programmieren, muss es aber nicht:
 - C++ ist eine sog. multi-paradigm-Sprache

- ☞ Abweichungen in syntaktischen Feinheiten

- ☞ semantische Unterschiede

```
Java:           Hello h = new Hello(....);           // reference !
                h.speak();
C++:            Hello h = Hello(....);             // value !
                h.speak();
```

Ein erster Blick: Hello World

- ☞ In C++ muss **jeder** verwendete Bezeichner zuvor (oder in der gleichen Klasse) deklariert werden!
(auch Bezeichner aus der Standard-Bibliothek)

```
#include <string> .... std::string
```

statt

→ /usr/include/g++/string

```
String (--->java.lang.String)
```

Ein erster Blick: Hello World

- ☞ In C++ gibt es globale Funktions- (Variablen- und Typ-) Deklarationen
- ☞ Es gibt geschachtelte Gültigkeitsbereiche (Klassen und namespaces) aber ohne implizite Abbildung auf eine hierarchische Verzeichnisstruktur
- ☞ Ein Compilerlauf behandelt **GENAU EINE** Quelldatei pro Aufruf! (.... **make** !)
- ☞ Dies entspricht dem klassischen Paradigma der Übersetzung von C-Programmen: ermöglicht Migration, Portierbarkeit, Unix-Konformität

Ein erster Blick: Hello World

- ☞ In C++ gibt es Zeiger, Felder sind de facto Zeiger - keine Objekte, **this** ist ein Zeiger !
- ☞ Konvention des Programmaufrufs ist etwas anders
- ☞ Virtualität ist explizit zu spezifizieren
- ☞ Es gibt sog. **Destruktoren**
- ☞ Syntax von Zugriffsrechten ist etwas anders

Der zweite Blick: Effizienz

Problem 1: integer - Arithmetik

$$3^{10^n}$$

(modulo int-overflow)

Problem 2: double - Arithmetik

$$e \sim \left(1 + \frac{1}{n}\right)^n$$

$$[n = 10e8, 10e9, 10e10]$$

Der zweite Blick: Effizienz

```
class i {  
    public static void main(String []s)  
    {  
        int i=1;  
  
        for(int n=1; n<=100000000; ++n)  
            i*=3;  
        System.out.println( i );  
    }  
}
```

Java

Der zweite Blick: Effizienz

```
#include <iostream>

class i {
public:
static void main()
    {
        int i=1;

        for(int n=1; n<=100000000; ++n)
            i*=3;
        std::cout << i << std::endl;
    }
};

int main() {
    i::main();
    return 0;
}
```

C++

Der zweite Blick: Effizienz

```
class d {  
  
    public static void main(String []s)  
    {  
        double e=1;  
  
        for(int n=1; n<=100000000; ++n)  
            e*=1.00000001;  
        System.out.println( e );  
    }  
}
```

Java

Der zweite Blick: Effizienz

```
#include <iostream>

class d {
public:
static void main()
    {
        double e=1;

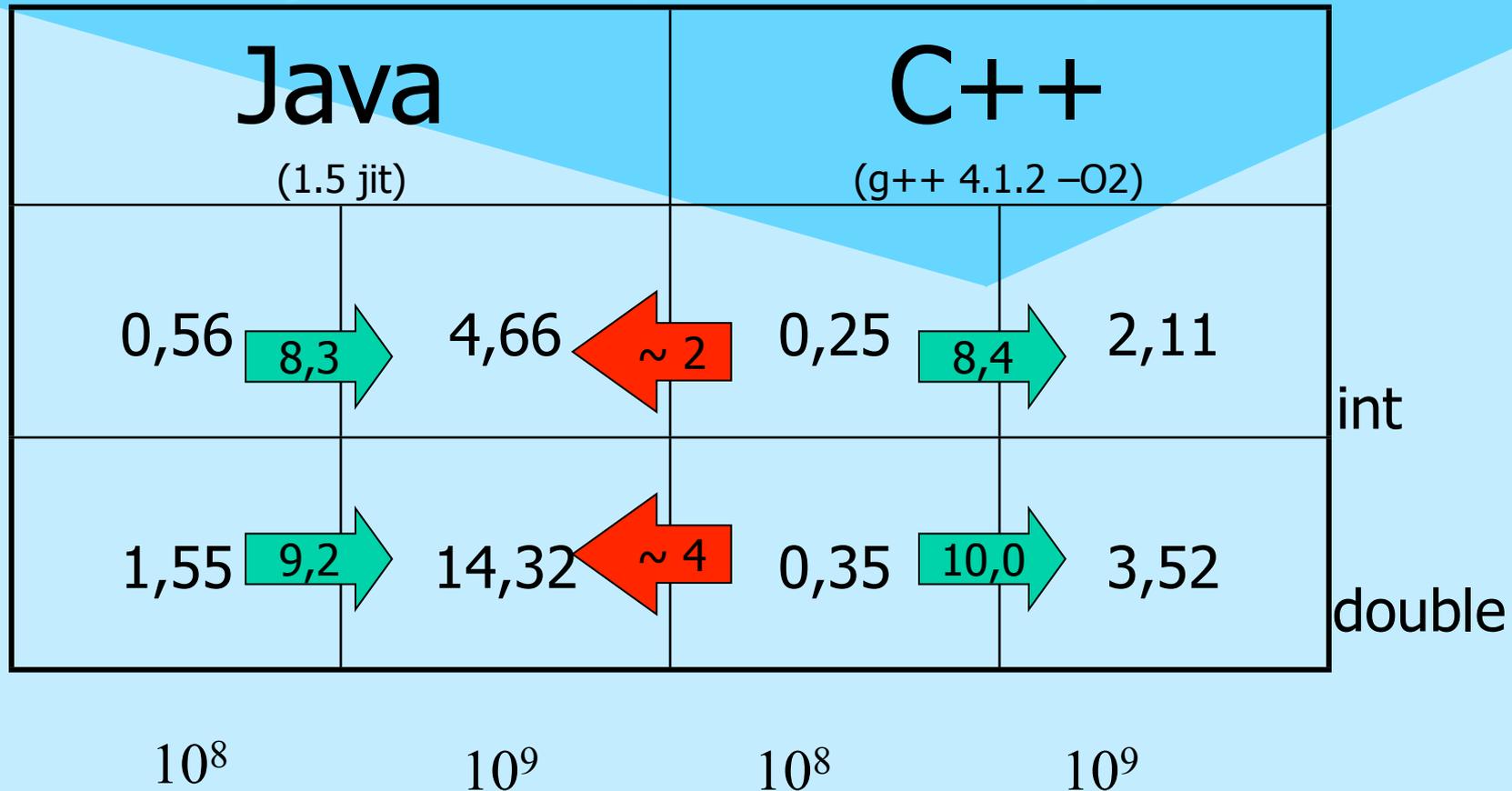
        for(int n=1; n<=100000000; ++n)
            e*=1.00000001;
        std::cout << e << std::endl;
    }
};

int main() {
    d::main();
    return 0; // not needed but good style
}
```

C++

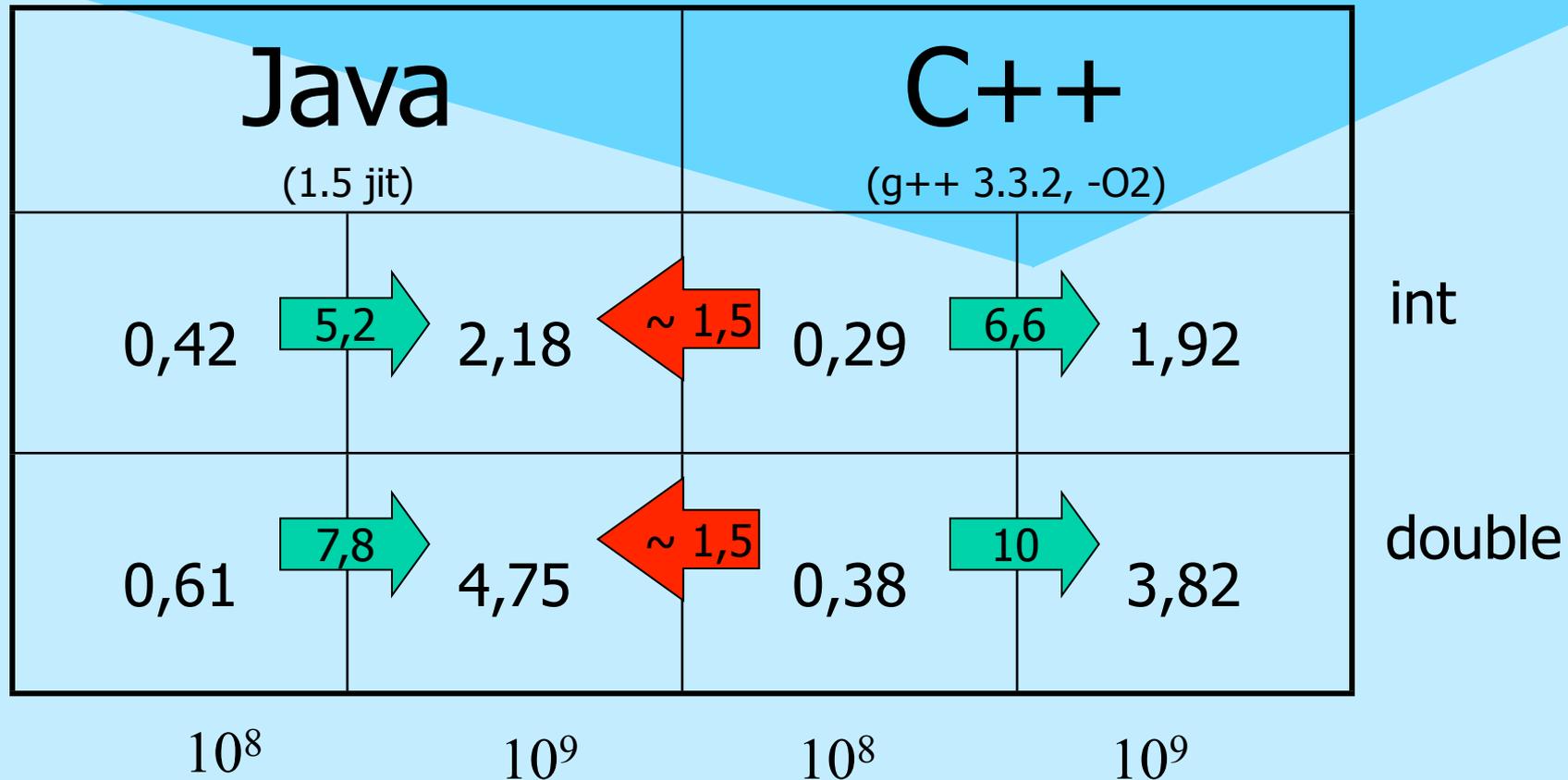
Der zweite Blick: Effizienz

Laufzeiten (Debian Linux, Pentium4 2 GHz)



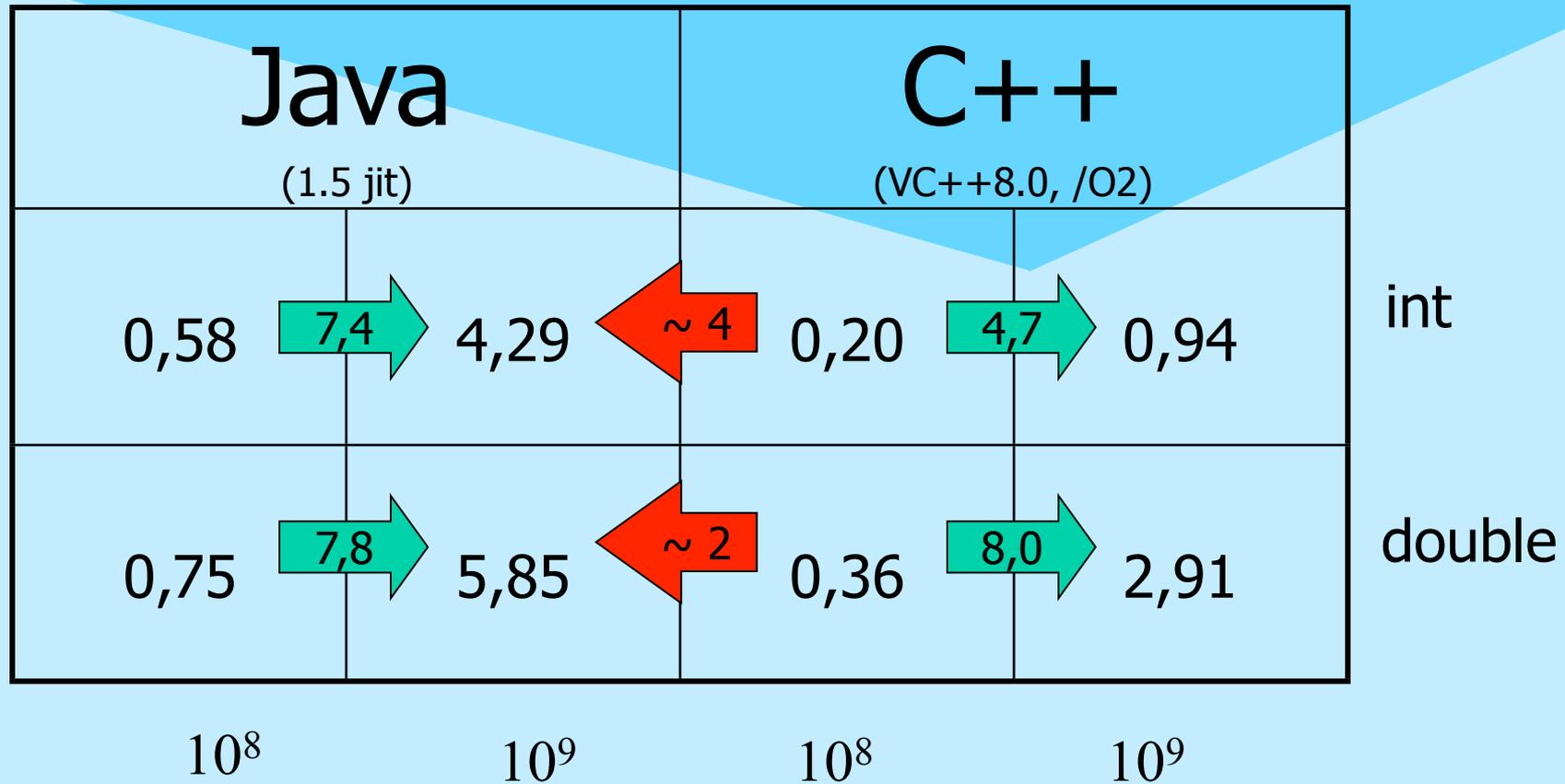
Der zweite Blick: Effizienz

Laufzeiten (Solaris 5.8, UltraSparc [8x]1050 MHz)



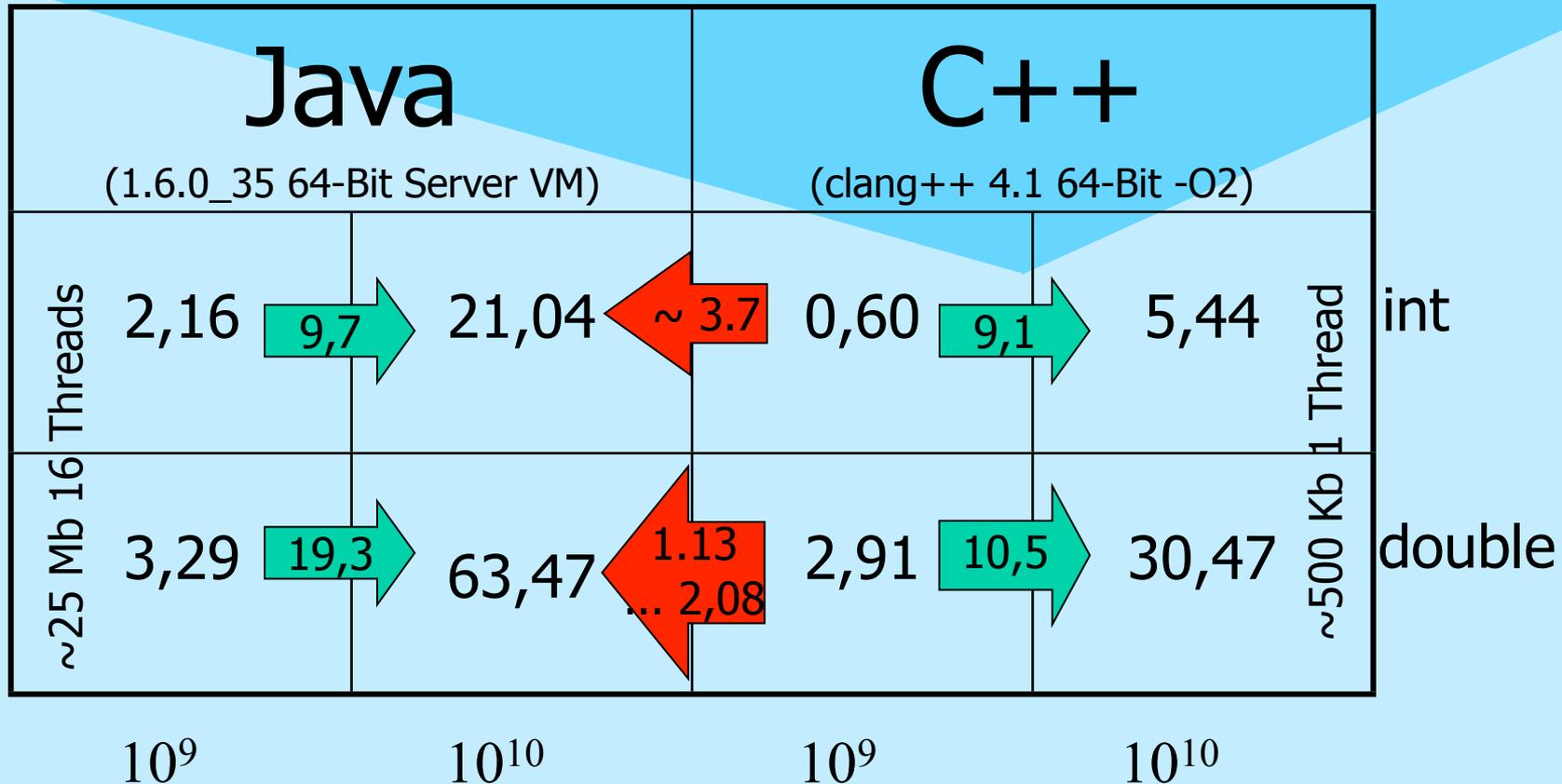
Der zweite Blick: Effizienz

Laufzeiten (Win XP, Pentium M 1,8 GHz)



Der zweite Blick: Effizienz (up to date :-)

Laufzeiten (Mac OS X 10.8.2, 1.86 GHz Intel Core 2 Duo)



C++ Historie

- Bjarne Stroustrup Ph.D. Arbeit 1978/79 an der Universität Cambridge: „Alternative Organisationsmöglichkeiten der Systemsoftware in verteilten Systemen“
- erste Implementation in Simula auf IBM360 (Simula67, NCC Oslo)
- Stroustrup: „Die Entwicklung des Simulators war das reinste Vergnügen, da Simula nahezu ideal für diesen Zweck erschien. Besonders beeindruckt wurde ich durch die Art, in der die Konzepte der Sprache mich beim Überdenken der Probleme meiner Anwendung unterstützten. Das Konzept der Klassen gestattete mir, die Konzepte meiner Anwendung direkt einzelnen Sprachkonstrukten zuzuordnen. So erhielt ich Programmcode, der in seiner Lesbarkeit allen Programmen anderer Sprachen überlegen war, die ich bisher gesehen hatte.“
- Simula - Compiler damals mit extrem schlechten Laufzeiteigenschaften

C++ Historie

- S.: „Um das Projekt nicht gänzlich abzubrechen - und Cambridge ohne Ph.D. zu verlassen -, schrieb ich den Simulator ein zweites Mal in BCPL Die Erfahrungen, die ich während des Entwickelns und der Fehlersuche in BCPL sammelte, waren grauenerregend.“
- erste Ideen zu C++ im Kontext von Untersuchungen Lastverteilung in UNIX-Netzen bei den Bell Labs Murray Hill, New Jersey: Stroustrup: „Ende 1979 hatte ich einen lauffähigen Präprozessor mit dem Namen Cpre geschrieben, der C um Simula-ähnliche Klassen erweiterte.“ -> C with classes

C++ Historie

