

Programmierpraktikum

Einfacher? Crosscompiler von Python nach C++

Felix Hensch
Julian Buchhorn

29. Dezember 2014

Betreuer
Dr. Holger Arndt

Inhaltsverzeichnis

1	Einleitung	3
2	Verwendung	5
3	Programme	5
4	Features	5
5	Codestruktur	5
	Abbildungsverzeichnis	6
	Tabellenverzeichnis	7
	Listings	8

1 Einleitung

Im Rahmen eines Programmierpraktikums an der bergischen Universität Wuppertal haben wir einen Compiler von Python zu C++ geschrieben. Dabei ging es in erster Linie darum, zu verstehen, wie ein Compiler funktioniert und nicht einen vollständigen Compiler zu schreiben.

Als vorläufiges Ziel haben wir uns gesetzt ein einfaches Testprogramm mit grundlegenden Syntaxelementen zu übersetzen (s. Lst. 1).

Listing 1: *Python Testprogramm für den Compiler*

```
1 from __future__ import division
2
3 """
4 Test program for codegeneration
5 """
6
7 def f(x):
8     """calculate"""
9     return x**2 + x*4 - (x-3) /x -1.3e2+2\
10         +5 #comment
11
12
13
14 if __name__ == '__main__':
15
16     L= [0]*10
17
18     for i in range(len(L)):
19         if i>3:
20             L[i]= f(i+1)
21         else:
22             L[i]+=1
23
24     for x in L:
25         print x
26
27     print 'program', 'end'
```

Das Ziel haben wir erreicht, unser Compiler kann in seinem momentanen Zustand das Python Programm übersetzen und liefert das C++ Programm in Listing 2. Dabei mussten wie erwartet einige Annahmen über das Programm gemacht werden um die Übersetzung zu vereinfachen.

Listing 2: *Vom Compiler erzeugtes C++ Programm*

```
1 #include <array>
2 #include <cmath>
3 #include <iostream>
4
5 using namespace std;
6
```

```
7  /*
8  Test program for codegeneration
9  */
10
11 double f(double x) {
12     /*calculate*/
13     return pow(x, 2)+ x*4 - (x-3) /(double) x-1.3e2+2
14         +5; //comment
15 }
16
17
18
19 int main() {
20
21     array<double,10> L={};
22
23     for (int i= 0; i<L.size(); i++) {
24         if (i>3) {
25             L[i]= f(i+1);
26         }
27         else {
28             L[i]++; // com
29         }
30     }
31
32     for (auto x : L) {
33         cout<< x << endl;
34     }
35
36     cout<< "program" << ' ' << "end" << endl;
37
38     return 0;
39
40 }
```

In den folgenden Kapiteln werden wir noch erklären wie man den Compiler verwenden kann, welche Programme man dafür benötigt und welche Features der Compiler im Moment unterstützt. Zum Schluss werden wir noch kurz auf die Struktur des Codes eingehen, den wir geschrieben haben.

Für eine detailliertere Erklärung wird auf die Bachelorarbeit von Felix Hensch verwiesen..., in welcher die Erstellung des Compilers fortgeführt wird.

3 Programme



5 Codestruktur

Abbildungsverzeichnis

1	Beispiel Parsetree	5
---	------------------------------	---

Tabellenverzeichnis

Listings

1	Python Testprogramm für den Compiler	3
2	Vom Compiler erzeugtes C++ Programm	3