Albert Ludwigs Universität Freiburg

Technische Fakultät

PicoC-Compiler

Übersetzung einer Untermenge von C in den Befehlssatz der RETI-CPU

Bachelorarbeit

Abgabedatum: 28th April 2022

Author:
Jürgen Mattheis

Gutachter:
Prof. Dr. Scholl

Betreung:
M.Sc. Seufert

Eine Bachelorarbeit am Lehrstuhl für
Betriebssysteme

ERKLÄRUNG

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Inhaltsverzeichnis

Abbildungsverzeichnis

Codeverzeichnis

Tabellenverzeichnis

Definitionen

1 Implementierung

1.1 Architektur

1.2 Lexikalische Analyse

1.2.1 Verwendung von Lark

1.2.2 Basic Parser

1.3 Syntaktische Analyse

1.3.1 Verwendung von Lark

1.3.2 Umsetzung von Präzidenz

Die **PicoC Sprache** hat dieselben **Präzidenzregeln** implementiert, wie die **Sprache C**[1]. Die **Präzidenzregeln** von **PicoC** sind in Tabelle 1.1 aufgelistet.

[1] *C Operator Precedence - cppreference.com.*

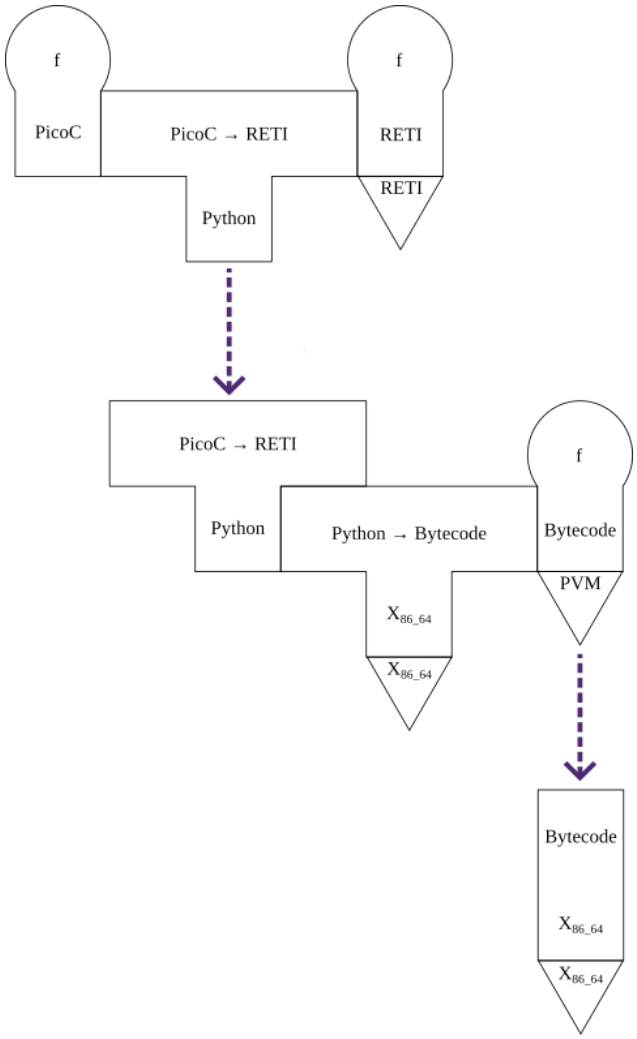

Abbildung 1.1: Cross-Compiler Kompiliervorgang ausgeschrieben

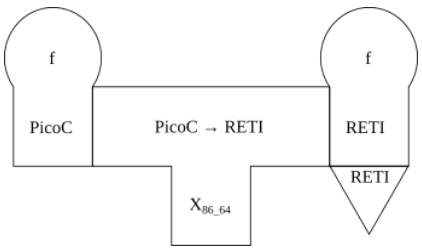

Abbildung 1.2: Cross-Compiler Kompiliervorgang Kurzform

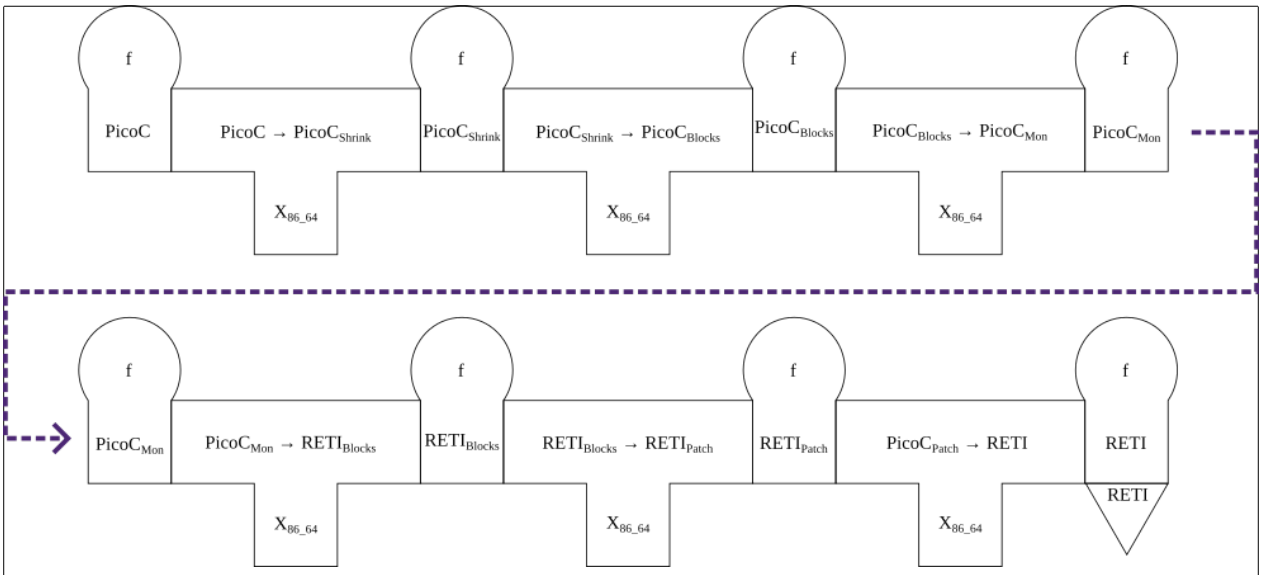

Abbildung 1.3: Architektur mit allen Passes ausgeschrieben

Präzidenz	Operator	Beschreibung	Assoziativität

1	`a()`
	`a[]`
	`a.b`
		Funktionsaufruf
		Indexzugriff
		Attributzugriff
			Links, dann rechts →

2	`-a`
	`!a ~a`
	`*a &a`
		Unäres Minus
		Logisches NOT und Bitweise NOT
		Dereferenz und Referenz, auch Adresse-von
			Rechts, dann links ←

3	`a*b a/b a%b`	Multiplikation, Division und Modulo	Links, dann rechts →
4	`a+b a-b`	Addition und Subtraktion	
5	`a<b a<=b` `a>b a>=b`	Kleiner, Kleiner Gleich, Größer, Größer gleich	
6	`a==b a!=b`	Gleichheit und Ungleichheit	
7	`a&b`	Bitweise UND	
8	`a^b`	Bitweise XOR (exclusive or)	
9	`a|b`	Bitweise ODER (inclusive or)	
10	`a&&b`	Logiches UND	
11	$a \| \| b$	Logisches ODER	
12	`a=b`	Zuweisung	Rechts, dann links ←
13	`a,b`	Komma	Links, dann rechts →

Tabelle 1.1: Präzidenzregeln von PicoC

1.3.3 Derivation Tree Generierung

1.3.4 Early Parser

1.3.5 Derivation Tree Vereinfachung

1.3.6 Abstrakt Syntax Tree Generierung

1.3.6.1 ASTNode

1.3.6.2 PicoC Nodes

1.3.6.3 RETI Nodes

1.4 Code Generierung

1.4.1 Passes

1.4.1.1 PicoC-Shrink Pass

1.4.1.2 PicoC-Blocks Pass

1.4.1.3 PicoC-Mon Pass

Definition 1.1: Symboltabelle

1.4.1.4 RETI-Blocks Pass

1.4.1.5 RETI-Patch Pass

1.4.1.6 RETI Pass

1.4.2 Umsetzung von Pointern

1.4.2.1 Referenzierung

```
1 void main() {
2   int var = 42;
3   int *pntr = &var;
4 }
```

Code 1.1: PicoC Code für Pointer Referenzierung

```
File
  Name './example_pntr_ref.ast',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Assign
          Alloc
            Writeable,
            IntType 'int',
            Name 'var',
          Num '42',
        Assign
          Alloc
            Writeable,
            PntrDecl
              Num '1',
              IntType 'int',
            Name 'pntr',
          Ref
            Name 'var'
      ]
  ]
```

Code 1.2: Abstract Syntax Tree für Pointer Referenzierung

```
File
  Name './example_pntr_ref.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        Exp
          Num '42',
        Assign
          GlobalWrite
            Num '0',
          Tmp
            Num '1',
        Ref
          GlobalRead
            Num '0',
        Assign
          GlobalWrite
            Num '1',
          Tmp
            Num '1',
        Return
          Empty
      ]
  ]
```

Code 1.3: PicoC Mon Pass für Pointer Referenzierung

```
File
  Name './example_pntr_ref.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 0,
        ADDI SP 1,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.4: RETI Blocks Pass für Pointer Referenzierung

1.4.2.2 Pointer Dereferenzierung durch Zugriff auf Arrayindex ersetzen

```
void main() {
  int var = 42;
  int *pntr = &var;
  *pntr;
}
```

Code 1.5: PicoC Code für Pointer Dereferenzierung

```
File
  Name './example_pntr_deref.ast',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Assign
          Alloc
            Writeable,
            IntType 'int',
            Name 'var',
          Num '42',
        Assign
          Alloc
            Writeable,
            PntrDecl
              Num '1',
              IntType 'int',
            Name 'pntr',
          Ref
            Name 'var',
        Exp
          Deref
            Name 'pntr',
            Num '0'
      ]
  ]
```

Code 1.6: Abstract Syntax Tree für Pointer Dereferenzierung

```
File
  Name './example_pntr_deref.picoc_shrink',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Assign
          Alloc
            Writeable,
            IntType 'int',
            Name 'var',
          Num '42',
        Assign
          Alloc
            Writeable,
            PntrDecl
              Num '1',
              IntType 'int',
            Name 'pntr',
          Ref
            Name 'var',
        Exp
          Subscr
            Name 'pntr',
            Num '0'
      ]
  ]
```

Code 1.7: PicoC Shrink Pass für Pointer Dereferenzierung

1.4.3 Umsetzung von Arrays

1.4.3.1 Initialisierung von Arrays

```
void main() {
  int ar[2][1] = {{4}, {2}};
}
```

Code 1.8: PicoC Code für Array Initialisierung

```
File
  Name './example_array_init.ast',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Assign
          Alloc
            Writeable,
            ArrayDecl
              [
                Num '2',
                Num '1'
              ],
              IntType 'int',
            Name 'ar',
          Array
            [
              Array
                [
                  Num '4'
                ],
              Array
                [
                  Num '2'
                ]
            ]
      ]
  ]
```

Code 1.9: Abstract Syntax Tree für Array Initialisierung

```
File
  Name './example_array_init.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        Exp
          Num '4',
        Exp
          Num '2',
        Assign
          GlobalWrite
            Num '0',
          Tmp
            Num '2',
        Return
          Empty
      ]
  ]
```

Code 1.10: PicoC Mon Pass für Array Initialisierung

```
File
  Name './example_array_init.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 4,
        STOREIN SP ACC 1,
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        LOADIN SP ACC 2,
        STOREIN DS ACC 0,
        ADDI SP 2,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.11: RETI Blocks Pass für Array Initialisierung

1.4.3.2 Zugriff auf Arrayindex

Der Zugriff auf einen bestimmten Index eines Arrays ist wie folgt umgesetzt:

```
void main() {
  int ar[2] = {1, 2};
  ar[2];
}
```

Code 1.12: PicoC Code für Zugriff auf Arrayindex

```
File
  Name './example_array_access.ast',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Assign
          Alloc
            Writeable,
            ArrayDecl
              [
                Num '2'
              ],
              IntType 'int',
            Name 'ar',
          Array
            [
              Num '1',
              Num '2'
            ],
        Exp
          Subscr
            Name 'ar',
            Num '2'
      ]
  ]
```

Code 1.13: Abstract Syntax Tree für Zugriff auf Arrayindex

```
File
  Name './example_array_access.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        Exp
          Num '1',
        Exp
          Num '2',
        Assign
          GlobalWrite
            Num '0',
          Tmp
            Num '2',
        Ref
          GlobalRead
            Num '0',
        Exp
          Num '2',
        Ref
          Subscr
            Tmp
              Num '2',
            Tmp
              Num '1',
        Exp
          Subscr
            Tmp
              Num '1',
            Num '0',
        Return
          Empty
      ]
  ]
```

Code 1.14: PicoC Mon Pass für Zugriff auf Arrayindex

```
File
  Name './example_array_access.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 1,
        STOREIN SP ACC 1,
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        LOADIN SP ACC 2,
        STOREIN DS ACC 0,
        ADDI SP 2,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        LOADIN SP IN1 2,
        LOADIN SP IN2 1,
        MULTI IN2 1,
        ADD IN1 IN2,
        ADDI SP 1,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        LOADIN IN1 ACC 0,
        STOREIN SP ACC 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.15: RETI Blocks Pass für Zugriff auf Arrayindex

1.4.3.3 Zuweisung an Arrayindex

```
void main() {
  int ar[2];
  ar[2] = 42;
}
```

Code 1.16: PicoC Code für Zuweisung an Arrayindex

```
File
  Name './example_array_assignment.ast',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Exp
          Alloc
            Writeable,
            ArrayDecl
              [
                Num '2'
              ],
            IntType 'int',
            Name 'ar',
        Assign
          Subscr
            Name 'ar',
            Num '2',
          Num '42'
      ]
  ]
```

Code 1.17: Abstract Syntax Tree für Zuweisung an Arrayindex

```
File
  Name './example_array_assignment.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        Exp
          Num '42',
        Ref
          GlobalRead
            Num '0',
        Exp
          Num '2',
        Ref
          Subscr
            Tmp
              Num '2',
            Tmp
              Num '1',
        Assign
          Subscr
            Tmp
              Num '1',
            Num '0',
          Tmp
            Num '2',
        Return
          Empty
      ]
  ]
```

Code 1.18: PicoC Mon Pass für Zuweisung an Arrayindex

```
File
  Name './example_array_assignment.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        LOADIN SP IN1 2,
        LOADIN SP IN2 1,
        MULTI IN2 1,
        ADD IN1 IN2,
        ADDI SP 1,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        LOADIN SP ACC 2,
        ADDI SP 2,
        STOREIN IN1 ACC 0,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.19: RETI Blocks Pass für Zuweisung an Arrayindex

1.4.4 Umsetzung von Structs

1.4.4.1 Deklaration von Structs

```
struct st1 {int *ar[3];};

struct st2 {struct st1 st;};

void main() {
}
```

Code 1.20: PicoC Code für Deklaration von Structs

```
SymbolTable
  [
    Symbol(
      {
```

```
5         type qualifier:         Empty()
6         datatype:               ArrayDecl([Num('3')], PntrDecl(Num('1'), IntType('int')))
7         name:                   Name('ar@st1')
8         value or address:       Empty()
9         position:               Pos(Num('1'), Num('17'))
10        size:                   Num('3')
11      },
12    Symbol(
13      {
14        type qualifier:         Empty()
15        datatype:               StructDecl(Name('st1'), [Alloc(Writeable(),
          ↪  ArrayDecl([Num('3')], PntrDecl(Num('1'), IntType('int'))), Name('ar'))])
16        name:                   Name('st1')
17        value or address:       [Name('ar@st1')]
18        position:               Pos(Num('1'), Num('7'))
19        size:                   Num('3')
20      },
21    Symbol(
22      {
23        type qualifier:         Empty()
24        datatype:               StructSpec(Name('st1'))
25        name:                   Name('st@st2')
26        value or address:       Empty()
27        position:               Pos(Num('3'), Num('23'))
28        size:                   Num('3')
29      },
30    Symbol(
31      {
32        type qualifier:         Empty()
33        datatype:               StructDecl(Name('st2'), [Alloc(Writeable(),
          ↪  StructSpec(Name('st1')), Name('st'))])
34        name:                   Name('st2')
35        value or address:       [Name('st@st2')]
36        position:               Pos(Num('3'), Num('7'))
37        size:                   Num('3')
38      },
39    Symbol(
40      {
41        type qualifier:         Empty()
42        datatype:               FunDecl(VoidType('void'), Name('main'), [])
43        name:                   Name('main')
44        value or address:       Empty()
45        position:               Pos(Num('5'), Num('5'))
46        size:                   Empty()
47      }
48  ]
```

Code 1.21: Symboltabelle für Deklaration von Structs

1.4.4.2 Initialisierung von Structs

```
1 struct st1 {int *pntr[1];};
2
3 struct st2 {struct st1 st;};
4
5 void main() {
6   int var = 42;
7   struct st1 st = {.st={.pntr={{&var}}}};
8 }
```

Code 1.22: PicoC Code für Initialisierung von Structs

```
1 File
2   Name './example_struct_init.ast',
3   [
4     StructDecl
5       Name 'st1',
6       [
7         Alloc
8           Writeable,
9           ArrayDecl
10            [
11              Num '1'
12            ],
13            PntrDecl
14              Num '1',
15              IntType 'int',
16          Name 'pntr'
17      ],
18    StructDecl
19      Name 'st2',
20      [
21        Alloc
22          Writeable,
23          StructSpec
24            Name 'st1',
25          Name 'st'
26      ],
27    FunDef
28      VoidType 'void',
29      Name 'main',
30      [],
31      [
32        Assign
33          Alloc
34            Writeable,
35            IntType 'int',
36            Name 'var',
37          Num '42',
38        Assign
39          Alloc
40            Writeable,
41            StructSpec
```

```
42              Name 'st1',
43            Name 'st',
44          Struct
45            [
46              Assign
47                Name 'st',
48                Struct
49                  [
50                    Assign
51                      Name 'pntr',
52                      Array
53                        [
54                          Array
55                            [
56                              Ref
57                                Name 'var'
58                            ]
59                        ]
60                  ]
61            ]
62      ]
63  ]
```

Code 1.23: Abstract Syntax Tree für Initialisierung von Structs

```
1 File
2   Name './example_struct_init.picoc_mon',
3   [
4     Block
5       Name 'main.0',
6       [
7         Exp
8           Num '42',
9         Assign
10          GlobalWrite
11            Num '0',
12          Tmp
13            Num '1',
14        Ref
15          GlobalRead
16            Num '0',
17        Assign
18          GlobalWrite
19            Num '1',
20          Tmp
21            Num '1',
22        Return
23          Empty
24      ]
25  ]
```

Code 1.24: PicoC Mon Pass für Initialisierung von Structs

```
File
  Name './example_struct_init.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 0,
        ADDI SP 1,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.25: RETI Blocks Pass für Initialisierung von Structs

1.4.4.3 Zugriff auf Structattribut

```
struct pos {int x; int y;};

void main() {
  struct pos st = {.x=4, .y=2};
  st.y;
}
```

Code 1.26: PicoC Code für Zugriff auf Structattribut

```
File
  Name './example_struct_attr_access.ast',
  [
    StructDecl
      Name 'pos',
      [
        Alloc
          Writeable,
          IntType 'int',
          Name 'x',
        Alloc
          Writeable,
          IntType 'int',
          Name 'y'
```

```
15      ],
16    FunDef
17      VoidType 'void',
18      Name 'main',
19      [],
20      [
21        Assign
22          Alloc
23            Writeable,
24            StructSpec
25              Name 'pos',
26            Name 'st',
27          Struct
28            [
29              Assign
30                Name 'x',
31                Num '4',
32              Assign
33                Name 'y',
34                Num '2'
35            ],
36        Exp
37          Attr
38            Name 'st',
39            Name 'y'
40      ]
41  ]
```

Code 1.27: Abstract Syntax Tree für Zugriff auf Structattribut

```
File
  Name './example_struct_attr_access.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        Exp
          Num '4',
        Exp
          Num '2',
        Assign
          GlobalWrite
            Num '0',
          Tmp
            Num '2',
        Ref
          GlobalRead
            Num '0',
        Ref
          Attr
            Tmp
              Num '1',
            Name 'y',
        Exp
          Subscr
            Tmp
              Num '1',
            Num '0',
        Return
          Empty
      ]
  ]
```

Code 1.28: PicoC Mon Pass für Zugriff auf Structattribut

```
File
  Name './example_struct_attr_access.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 4,
        STOREIN SP ACC 1,
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        LOADIN SP ACC 2,
        STOREIN DS ACC 0,
        ADDI SP 2,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        ADDI IN1 1,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        LOADIN IN1 ACC 0,
        STOREIN SP ACC 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.29: RETI Blocks Pass für Zugriff auf Structattribut

1.4.4.4 Zuweisung an Structattribut

```
struct pos {int x; int y;};

void main() {
  struct pos st = {.x=4, .y=2};
  st.y = 42;
}
```

Code 1.30: PicoC Code für Zuweisung an Structattribut

```
File
  Name './example_struct_attr_assignment.ast',
  [
    StructDecl
      Name 'pos',
      [
```

```
      Alloc
        Writeable,
        IntType 'int',
        Name 'x',
      Alloc
        Writeable,
        IntType 'int',
        Name 'y'
    ],
  FunDef
    VoidType 'void',
    Name 'main',
    [],
    [
      Assign
        Alloc
          Writeable,
          StructSpec
            Name 'pos',
          Name 'st',
        Struct
          [
            Assign
              Name 'x',
              Num '4',
            Assign
              Name 'y',
              Num '2'
          ],
      Assign
        Attr
          Name 'st',
          Name 'y',
        Num '42'
    ]
]
```

Code 1.31: Abstract Syntax Tree für Zuweisung an Structattribut

```
File
 Name './example_struct_attr_assignment.picoc_mon',
 [
   Block
     Name 'main.0',
     [
       Exp
         Num '4',
       Exp
         Num '2',
       Assign
         GlobalWrite
           Num '0',
         Tmp
           Num '2',
       Exp
         Num '42',
       Ref
         GlobalRead
           Num '0',
       Ref
         Attr
           Tmp
             Num '1',
           Name 'y',
       Assign
         Subscr
           Tmp
             Num '1',
           Num '0',
         Tmp
           Num '2',
       Return
         Empty
     ]
 ]
```

Code 1.32: PicoC Mon Pass für Zuweisung an Structattribut

```
File
  Name './example_struct_attr_assignment.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 4,
        STOREIN SP ACC 1,
        SUBI SP 1,
        LOADI ACC 2,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        LOADIN SP ACC 2,
        STOREIN DS ACC 0,
        ADDI SP 2,
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        ADDI IN1 1,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        LOADIN SP ACC 2,
        ADDI SP 2,
        STOREIN IN1 ACC 0,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.33: RETI Blocks Pass für Zuweisung an Structattribut

1.4.5 Umsetzung des Zusammenspiels der Derived Datatypes

1.4.5.1 Definition von Variablen mit den Derived Datatypes

```c
struct ar_with_len {int ar[2]; int len;};

void main() {
  struct ar_with_len st_ar[3];
  int (*pntr1)[3];
  int *(*pntr2)[3];
}
```

Code 1.34: PicoC Code für Definition von Variablen

```
SymbolTable
  [
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               ArrayDecl([Num('2')], IntType('int'))
        name:                   Name('ar@ar_with_len')
        value or address:       Empty()
        position:               Pos(Num('1'), Num('24'))
        size:                   Num('2')
      },
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               IntType('int')
        name:                   Name('len@ar_with_len')
        value or address:       Empty()
        position:               Pos(Num('1'), Num('35'))
        size:                   Num('1')
      },
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               StructDecl(Name('ar_with_len'), [Alloc(Writeable(),
        ↪  ArrayDecl([Num('2')], IntType('int')), Name('ar')), Alloc(Writeable(),
        ↪  IntType('int'), Name('len'))])
        name:                   Name('ar_with_len')
        value or address:       [Name('ar@ar_with_len'), Name('len@ar_with_len')]
        position:               Pos(Num('1'), Num('7'))
        size:                   Num('3')
      },
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               FunDecl(VoidType('void'), Name('main'), [])
        name:                   Name('main')
        value or address:       Empty()
        position:               Pos(Num('3'), Num('5'))
        size:                   Empty()
      },
    Symbol(
      {
        type qualifier:         Writeable()
        datatype:               ArrayDecl([Num('3')], StructSpec(Name('ar_with_len')))
        name:                   Name('st_ar@main')
        value or address:       Num('0')
        position:               Pos(Num('4'), Num('21'))
        size:                   Num('9')
      },
    Symbol(
      {
        type qualifier:         Writeable()
        datatype:               PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
        name:                   Name('pntr1@main')
        value or address:       Num('9')
        position:               Pos(Num('5'), Num('8'))
        size:                   Num('1')
```

```
56      },
57    Symbol(
58      {
59        type qualifier:         Writeable()
60        datatype:               PntrDecl(Num('1'), ArrayDecl([Num('3')], PntrDecl(Num('1'),
          ↪  IntType('int'))))
61        name:                   Name('pntr2@main')
62        value or address:       Num('10')
63        position:               Pos(Num('6'), Num('9'))
64        size:                   Num('1')
65      }
66  ]
```

Code 1.35: Symboltabelle für Definition von Variablen

1.4.5.2 Zugriff auf Variablen mit Derived Datatypes

```
1 struct st1 {int (*ar)[1];};
2
3 void main() {
4   int var[1] = {42};
5   struct st1 st_first = {.ar=&var};
6   (*st_first.ar)[0];
7 }
```

Code 1.36: PicoC Code für Zugriff auf Variablen mit Derived Datatypes

```
1 File
2   Name './example_derived_dts_combined.ast',
3   [
4     StructDecl
5       Name 'st1',
6       [
7         Alloc
8           Writeable,
9           PntrDecl
10            Num '1',
11            ArrayDecl
12              [
13                Num '1'
14              ],
15              IntType 'int',
16          Name 'ar'
17      ],
18    FunDef
19      VoidType 'void',
20      Name 'main',
21      [],
22      [
23        Assign
24          Alloc
25            Writeable,
```

```
26            ArrayDecl
27              [
28                Num '1'
29              ],
30              IntType 'int',
31            Name 'var',
32          Array
33            [
34              Num '42'
35            ],
36        Assign
37          Alloc
38            Writeable,
39            StructSpec
40              Name 'st1',
41            Name 'st_first',
42          Struct
43            [
44              Assign
45                Name 'ar',
46                Ref
47                  Name 'var'
48            ],
49        Exp
50          Subscr
51            Deref
52              Attr
53                Name 'st_first',
54                Name 'ar',
55              Num '0',
56            Num '0'
57      ]
58  ]
```

Code 1.37: Abstract Syntax Tree für Zugriff auf Variablen mit Derived Datatypes

```
File
  Name './example_derived_dts_combined.picoc_mon',
  [
    Block
      Name 'main.0',
      [
        Exp
          Num '42',
        Assign
          GlobalWrite
            Num '0',
          Tmp
            Num '1',
        Ref
          GlobalRead
            Num '0',
        Assign
          GlobalWrite
            Num '1',
          Tmp
            Num '1',
        Ref
          GlobalRead
            Num '1',
        Ref
          Attr
            Tmp
              Num '1',
            Name 'ar',
        Exp
          Num '0',
        Ref
          Subscr
            Tmp
              Num '2',
            Tmp
              Num '1',
        Exp
          Num '0',
        Ref
          Subscr
            Tmp
              Num '2',
            Tmp
              Num '1',
        Exp
          Subscr
            Tmp
              Num '1',
            Num '0',
        Return
          Empty
      ]
  ]
```

Code 1.38: PicoC Mon Pass für Zugriff auf Variablen mit Derived Datatypes

```
File
  Name './example_derived_dts_combined.reti_blocks',
  [
    Block
      Name 'main.0',
      [
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 0,
        ADDI SP 1,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        LOADIN SP ACC 1,
        STOREIN DS ACC 1,
        ADDI SP 1,
        SUBI SP 1,
        LOADI IN1 1,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        ADDI IN1 0,
        STOREIN SP IN1 1,
        SUBI SP 1,
        LOADI ACC 0,
        STOREIN SP ACC 1,
        LOADIN SP IN2 2,
        LOADIN IN2 IN1 0,
        LOADIN SP IN2 1,
        MULTI IN2 1,
        ADD IN1 IN2,
        ADDI SP 1,
        STOREIN SP IN1 1,
        SUBI SP 1,
        LOADI ACC 0,
        STOREIN SP ACC 1,
        LOADIN SP IN1 2,
        LOADIN SP IN2 1,
        MULTI IN2 1,
        ADD IN1 IN2,
        ADDI SP 1,
        STOREIN SP IN1 1,
        LOADIN SP IN1 1,
        LOADIN IN1 ACC 0,
        STOREIN SP ACC 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.39: RETI Blocks Pass für Zugriff auf Variablen mit Derived Datatypes

1.4.6 Umsetzung von Funktionen

1.4.6.1 Funktionen auflösen zu RETI Code

```
1  void main() {
2    return;
3  }
4
5  void fun1() {
6  }
7
8  int fun2() {
9    return 1;
10 }
```

Code 1.40: PicoC Code für 3 Funktionen

```
1  File
2    Name './example_3_funs.ast',
3    [
4      FunDef
5        VoidType 'void',
6        Name 'main',
7        [],
8        [
9          Return
10           Empty
11       ],
12     FunDef
13       VoidType 'void',
14       Name 'fun1',
15       [],
16       [],
17     FunDef
18       IntType 'int',
19       Name 'fun2',
20       [],
21       [
22         Return
23           Num '1'
24       ]
25   ]
```

Code 1.41: Abstract Syntax Tree für 3 Funktionen

```
File
  Name './example_3_funs.picoc_blocks',
  [
    FunDef
      VoidType 'void',
      Name 'main',
      [],
      [
        Block
          Name 'main.2',
          [
            Return
              Empty
          ]
      ],
    FunDef
      VoidType 'void',
      Name 'fun1',
      [],
      [
        Block
          Name 'fun1.1',
          []
      ],
    FunDef
      IntType 'int',
      Name 'fun2',
      [],
      [
        Block
          Name 'fun2.0',
          [
            Return
              Num '1'
          ]
      ]
  ]
```

Code 1.42: PicoC Blocks Pass für 3 Funktionen

```
File
  Name './example_3_funs.picoc_mon',
  [
    Block
      Name 'main.2',
      [
        Return
          Empty
      ],
    Block
      Name 'fun1.1',
      [
        Return
          Empty
      ],
    Block
      Name 'fun2.0',
      [
        Exp
          Num '1',
        Return
          Tmp
            Num '1'
      ]
  ]
```

Code 1.43: PicoC Mon Pass für 3 Funktionen

```
File
  Name './example_3_funs.reti_blocks',
  [
    Block
      Name 'main.2',
      [
        LOADIN BAF PC -1
      ],
    Block
      Name 'fun1.1',
      [
        LOADIN BAF PC -1
      ],
    Block
      Name 'fun2.0',
      [
        SUBI SP 1,
        LOADI ACC 1,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.44: RETI Blocks Pass für 3 Funktionen

1.4.6.1.1 Sprung zur Main Funktion

```
1 void fun1() {
2 }
3
4 int fun2() {
5   return 1;
6 }
7
8 void main() {
9   return;
10 }
```

Code 1.45: PicoC Code für Funktionen, wobei die main Funktion nicht die erste Funktion ist

```
1 File
2   Name './example_3_funs_main.picoc_mon',
3   [
4     Block
5       Name 'fun1.2',
6       [
7         Return
8           Empty
9       ],
10    Block
11      Name 'fun2.1',
12      [
13        Exp
14          Num '1',
15        Return
16          Tmp
17            Num '1'
18      ],
19    Block
20      Name 'main.0',
21      [
22        Return
23          Empty
24      ]
25  ]
```

Code 1.46: PicoC Mon Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist

```
File
  Name './example_3_funs_main.reti_blocks',
  [
    Block
      Name 'fun1.2',
      [
        LOADIN BAF PC -1
      ],
    Block
      Name 'fun2.1',
      [
        SUBI SP 1,
        LOADI ACC 1,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ],
    Block
      Name 'main.0',
      [
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.47: PicoC Blocks Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist

```
File
  Name './example_3_funs_main.reti_patch',
  [
    Block
      Name 'start.3',
      [
        Exp
          GoTo
            Name 'main.0'
      ],
    Block
      Name 'fun1.2',
      [
        LOADIN BAF PC -1
      ],
    Block
      Name 'fun2.1',
      [
        SUBI SP 1,
        LOADI ACC 1,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ],
    Block
      Name 'main.0',
      [
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.48: PicoC Patch Pass für Funktionen, wobei die main Funktion nicht die erste Funktion ist

1.4.6.2 Funktionsdeklaration und -definition

```
int fun2(int var);

void fun1() {
}

void main() {
  fun1();
  fun2(42);
  return;
}

int fun2(int var) {
  return var;
}
```

Code 1.49: PicoC Code für Funktionen, wobei eine Funktion vorher deklariert werden muss

```
SymbolTable
  [
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               FunDecl(IntType('int'), Name('fun2'), [Alloc(Writeable(),
        ↪  IntType('int'), Name('var'))])
        name:                   Name('fun2')
        value or address:       Empty()
        position:               Pos(Num('1'), Num('4'))
        size:                   Empty()
      },
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               FunDecl(VoidType('void'), Name('fun1'), [])
        name:                   Name('fun1')
        value or address:       Empty()
        position:               Pos(Num('3'), Num('5'))
        size:                   Empty()
      },
    Symbol(
      {
        type qualifier:         Empty()
        datatype:               FunDecl(VoidType('void'), Name('main'), [])
        name:                   Name('main')
        value or address:       Empty()
        position:               Pos(Num('6'), Num('5'))
        size:                   Empty()
      },
    Symbol(
      {
        type qualifier:         Writeable()
        datatype:               IntType('int')
        name:                   Name('var@fun2')
        value or address:       Num('0')
        position:               Pos(Num('12'), Num('13'))
        size:                   Num('1')
      }
  ]
```

Code 1.50: Symboltabelle für Funktionen, wobei eine Funktion vorher deklariert werden muss

1.4.6.2.1 Allocation von Variablen

1.4.6.3 Funktionsaufruf

1.4.6.3.1 Ohne Rückgabewert

```
struct st {int attr1; int attr2[2];};

void stack_fun(struct st param[2][3]);

void main() {
  struct st local_var[2][3];
  stack_fun(local_var);
  return;
}

void stack_fun(struct st param[2][3]) {
  int local_var;
}
```

Code 1.51: PicoC Code für Funktionsaufruf ohne Rückgabewert

```
File
  Name './example_fun_call_no_return_value.picoc_mon',
  [
    Block
      Name 'main.1',
      [
        StackMalloc
          Num '2',
        Ref
          GlobalRead
            Num '0',
        NewStackframe
          Name 'stack_fun',
          GoTo
            Name 'addr@next_instr',
        Exp
          GoTo
            Name 'stack_fun.0',
        RemoveStackframe,
        Return
          Empty
      ],
    Block
      Name 'stack_fun.0',
      [
        Return
          Empty
      ]
  ]
```

Code 1.52: PicoC Mon Pass für Funktionsaufruf ohne Rückgabewert

```
File
  Name './example_fun_call_no_return_value.reti_blocks',
  [
    Block
      Name 'main.1',
      [
        SUBI SP 2,
        SUBI SP 1,
        LOADI IN1 0,
        ADD IN1 DS,
        STOREIN SP IN1 1,
        MOVE BAF ACC,
        ADDI SP 3,
        MOVE SP BAF,
        SUBI SP 4,
        STOREIN BAF ACC 0,
        LOADI ACC GoTo
                    Name 'addr@next_instr',
        ADD ACC CS,
        STOREIN BAF ACC -1,
        Exp
          GoTo
            Name 'stack_fun.0',
        MOVE BAF IN1,
        LOADIN IN1 BAF 0,
        MOVE IN1 SP,
        LOADIN BAF PC -1
      ],
    Block
      Name 'stack_fun.0',
      [
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.53: RETI Blocks Pass für Funktionsaufruf ohne Rückgabewert

```
SUBI SP 2;
SUBI SP 1;
LOADI IN1 0;
ADD IN1 DS;
STOREIN SP IN1 1;
MOVE BAF ACC;
ADDI SP 3;
MOVE SP BAF;
SUBI SP 4;
STOREIN BAF ACC 0;
LOADI ACC 14;
ADD ACC CS;
STOREIN BAF ACC -1;
JUMP 5;
MOVE BAF IN1;
LOADIN IN1 BAF 0;
MOVE IN1 SP;
LOADIN BAF PC -1;
LOADIN BAF PC -1;
```

Code 1.54: RETI Pass für Funktionsaufruf ohne Rückgabewert

1.4.6.3.2 Mit Rückgabewert

```
void stack_fun() {
  return 42;
}

void main() {
  int var = stack_fun();
}
```

Code 1.55: PicoC Code für Funktionsaufruf mit Rückgabewert

```
File
  Name './example_fun_call_with_return_value.picoc_mon',
  [
    Block
      Name 'stack_fun.1',
      [
        Exp
          Num '42',
        Return
          Tmp
            Num '1'
      ],
    Block
      Name 'main.0',
      [
        StackMalloc
          Num '2',
        NewStackframe
          Name 'stack_fun',
          GoTo
            Name 'addr@next_instr',
        Exp
          GoTo
            Name 'stack_fun.1',
        RemoveStackframe,
        Assign
          GlobalWrite
            Num '0',
          Tmp
            Num '1',
        Return
          Empty
      ]
  ]
```

Code 1.56: PicoC Mon Pass für Funktionsaufruf mit Rückgabewert

```
File
  Name './example_fun_call_with_return_value.reti_blocks',
  [
    Block
      Name 'stack_fun.1',
      [
        SUBI SP 1,
        LOADI ACC 42,
        STOREIN SP ACC 1,
        LOADIN SP ACC 1,
        ADDI SP 1,
        LOADIN BAF PC -1
      ],
    Block
      Name 'main.0',
      [
        SUBI SP 2,
        MOVE BAF ACC,
        ADDI SP 2,
        MOVE SP BAF,
        SUBI SP 2,
        STOREIN BAF ACC 0,
        LOADI ACC GoTo
                    Name 'addr@next_instr',
        ADD ACC CS,
        STOREIN BAF ACC -1,
        Exp
          GoTo
            Name 'stack_fun.1',
        MOVE BAF IN1,
        LOADIN IN1 BAF 0,
        MOVE IN1 SP,
        LOADIN SP ACC 1,
        STOREIN DS ACC 0,
        ADDI SP 1,
        LOADIN BAF PC -1
      ]
  ]
```

Code 1.57: RETI Blocks Pass für Funktionsaufruf mit Rückgabewert

```
JUMP 7;
SUBI SP 1;
LOADI ACC 42;
STOREIN SP ACC 1;
LOADIN SP ACC 1;
ADDI SP 1;
LOADIN BAF PC -1;
SUBI SP 2;
MOVE BAF ACC;
ADDI SP 2;
MOVE SP BAF;
SUBI SP 2;
STOREIN BAF ACC 0;
LOADI ACC 17;
ADD ACC CS;
STOREIN BAF ACC -1;
JUMP -15;
MOVE BAF IN1;
LOADIN IN1 BAF 0;
MOVE IN1 SP;
LOADIN SP ACC 1;
STOREIN DS ACC 0;
ADDI SP 1;
LOADIN BAF PC -1;
```

Code 1.58: RETI Pass für Funktionsaufruf mit Rückgabewert

1.4.6.3.3 Umsetzung von Call by Sharing für Arrays

```
int stack_fun(int (*param1)[3], int param2[2][3]) {
  int local_var;
}

void main() {
  int local_var1[2][3];
  int local_var2[2][3];
  stack_fun(local_var1, local_var2);
}
```

Code 1.59: PicoC Code für eine Funktionsdefinition

```
SymbolTable
  [
    Symbol(
      {
        type qualifier:         Empty()
```

```
6         datatype:               FunDecl(IntType('int'), Name('stack_fun'),
          ↪  [Alloc(Writeable(), PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int'))),
          ↪  Name('param1')), Alloc(Writeable(), ArrayDecl([Num('3')], IntType('int')),
          ↪  Name('param2'))])
7         name:                   Name('stack_fun')
8         value or address:       Empty()
9         position:               Pos(Num('1'), Num('4'))
10        size:                   Empty()
11      },
12    Symbol(
13      {
14        type qualifier:         Writeable()
15        datatype:               PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
16        name:                   Name('param1@stack_fun')
17        value or address:       Num('0')
18        position:               Pos(Num('1'), Num('20'))
19        size:                   Num('1')
20      },
21    Symbol(
22      {
23        type qualifier:         Writeable()
24        datatype:               PntrDecl(Num('1'), ArrayDecl([Num('3')], IntType('int')))
25        name:                   Name('param2@stack_fun')
26        value or address:       Num('1')
27        position:               Pos(Num('1'), Num('36'))
28        size:                   Num('1')
29      },
30    Symbol(
31      {
32        type qualifier:         Writeable()
33        datatype:               IntType('int')
34        name:                   Name('local_var@stack_fun')
35        value or address:       Num('2')
36        position:               Pos(Num('2'), Num('6'))
37        size:                   Num('1')
38      },
39    Symbol(
40      {
41        type qualifier:         Empty()
42        datatype:               FunDecl(VoidType('void'), Name('main'), [])
43        name:                   Name('main')
44        value or address:       Empty()
45        position:               Pos(Num('5'), Num('5'))
46        size:                   Empty()
47      },
48    Symbol(
49      {
50        type qualifier:         Writeable()
51        datatype:               ArrayDecl([Num('2'), Num('3')], IntType('int'))
52        name:                   Name('local_var1@main')
53        value or address:       Num('0')
54        position:               Pos(Num('6'), Num('6'))
55        size:                   Num('6')
56      },
57    Symbol(
58      {
59        type qualifier:         Writeable()
```

```
60          datatype:               ArrayDecl([Num('2'), Num('3')], IntType('int'))
61          name:                   Name('local_var2@main')
62          value or address:       Num('6')
63          position:               Pos(Num('7'), Num('6'))
64          size:                   Num('6')
65        }
66    ]
```

Code 1.60: Symboltabelle für eine Funktionsdefinition

1.4.6.3.4 Umsetzung von Call by Value für Structs

1.4.7 Umsetzung kleinerer Details

1.5 Fehlermeldungen

1.5.1 Error Handler

1.5.2 Arten von Fehlermeldungen

1.5.2.1 Syntaxfehler

1.5.2.2 Laufzeitfehler

Literatur

Online

- *C Operator Precedence - cppreference.com*. URL: https://en.cppreference.com/w/c/language/operator_precedence (besucht am 27. 04. 2022).