Hausarbeit von Carlotta Williams Lopez
Informatik
Dr. Hermann Lenhart
Proseminar
31.08.2021

Fortran
die Programmiersprache

—

Inhaltsangabe

1. Allgemein

1.1 Eigenschaften von Fortran

Fortran ist eine höhere Programmiersprache, die das Konzept der starken Typisierung verwendet, unter anderem kennt Fortran die explizite und implizite Typisierung.

Die Programmiersprache ist für numerische Berechnungen vorgesehen und wurde dafür über die Jahre hinweg optimiert. Im Gegensatz zu anderen Programmiersprachen hatte Fortran den Potenz-Operator ** von Anfang an, welches sich später sehr verbreitet hat. Außerdem sind bei Fortran Vektor- und Matrix-Operationen standardisiert.

Was sich auch als sehr nützlich ergibt, sind die unterschiedlichen Bibliotheken die bei Fortran möglich sind, diese sind nicht nur sehr umfangreich, sondern ermöglichen es für den Programmierer seine bevorzugte Speicherung der Programme zu benutzen, welches für Wissenschaftliche und numerische Berechnung praktisch ist.

2. Geschichte

Die Geschichte von Fortran geht bis auf das Jahr 1953 zurück, in diesem Jahr hat John W. Backus ein Programmierer bei IBM einen Vorschlag bei seinem Vorgesetzten eingereicht.

Dieser Vorschlag führte dazu das ein IBM-Team einen Compiler entwickeln sollte, dies natürlich unter der Leitung von John W. Backus. Ein Jahr später, dementsprechend 1954 begann das Projekt, es war auf sechs Monate ausgelegt. Harlan Herrick war es möglich am 20. September 1954 das erste Fortran-Programm auszuführen. Jedoch war der Compiler noch nicht Marktreif und wurde erst 1957 auf den Markt gebracht.

Der Compiler wurde mit jedem IBM 704-System ausgeliefert. Außerdem wurde der Compiler von Anfang an mit der Fähigkeit zur Optimierung ausgestattet, darauf hatte Backus bestanden.

3. Versionen

Fortran existiert nun über einen längeren Zeitraum, dies bedeutet das über die Jahre immer wieder Veränderung angenommen wurden.

Fortran beziehungsweise Fortran I ist das erste Fortran welches auf dem Markt 1957 erschien, dieses war jedoch nicht perfekt, denn schon 1957/58 wurde Fortran II veröffentlicht, hier wurden Inline-Assembler, Kommentare und andere Funktionen hinzugefügt.

1958 entstand Fortran III, bei dieser Version wurden kleine Änderungen vorgenommen, sie wurde jedoch nie offiziell freigegeben.

Fortran IV wurde 1961/62 veröffentlicht, dies war eine erweiterte und verbesserte Version der Vorgänger.

Vier Jahre später, dementsprechend 1966 wurde Fortran 66 herausgebracht. Fortran 66 war die erste standardisierte Fortran-Version und die erste standardisierte höhere Programmiersprache. Außerdem wurde Fortran ab hier aus von einer internationalen Organisation standardisiert.

1987 erschien Fortran 77 auf dem Markt, hie wurde die Do-Schleife, If Then-else, IF-ELSE-END IF, CHARACTER-Datentyp und andere Funktionen hinzugefügt.

Fortran 90 welches 1991 veröffentlicht wurde, wurde der free form style. Module, Zeiger, Datenverbund und vieles mehr hinzugefügt. Hinzufügen sollte Fortran 90, früher als Fortran 8x schon 1982 erscheinen, jedoch gab es Komplikationen und wurde auf das Jahr 1985 verschoben, danach wieder verschoben bis es schließlich 1991 geklappt hatte.

Fortran 95 welches 1997 erschien, hatte im Gegensatz zu Fortran 90 nur kleinere Änderungen.

2003 erschien Fortran 2003 welches OOP(Objektorientiertes Programmieren), C.Bindungen normalisierte.

Sieben Jahre, also 2010 erschien Fortran 2008 welches mit Co-Arrays und kleinere Änderungen erschien.

Die neueste Version die von Fortran existiert ist Fortran 2018, welches 2018 herausgebracht wurde, hier wurden nur kleine Änderungen vorgenommen.

4. Bibliotheken

Es gibt drei unterschiedliche Möglichkeiten eine Bibliothek in Fortran zu führen. Dies wären die dynamische Bibliothek, statische Bibliothek und die Quelltext-Bibliothek.

4.1 Dynamische Bibliothek

Bei einer dynamischen Bibliothek wird diese erst bei Bedarf in den Arbeitsspeicher geladen. Sie werden mit sogenannten Lader mit dem ausführbaren Programm verbunden. Dies ist sehr vom Vorteil, da dadurch die Bibliothek nur einmal im Speicher gehalten werden muss, auch wenn sie von mehreren Programmen gleichzeitig genutzt wird.

Sehr Vorteilhaft ist dieses bei Multitasking-Systemen, da wenn die Bibliothek insgesamt sehr groß ist und von mehreren Prozessen gleichzeitig verwendet wird. Hier wird die Bibliothek bei ihrer ersten Verwendung in den Speicher geladen. Sollte es dazu kommen das ein Programm auf ein Unterprogramm verweist, das noch nicht eingebunden wurde, so wird ein Laufzeitbinder aktiviert. Welcher dann im Speicher die vorhandene Bibliothek das Unterprogramm sucht, wenn er das Unterprogramm findet, fügt er die Adresse am Aufruf Punkt ein und führt das Unterprogramm erstmalig aus.

Dies führt dazu, dass falls das Unterprogramm des Weiteren aufgerufen wird, die Adresse schon vorhanden ist und dann direkt aufgerufen werden kann. Leider ist die Ausführungszeit, insbesondere die Startzeit eines Programms, ist hier geringfügig erhöht. Dies wird jedoch in Kauf genommen, da der Programmcode der Bibliotheksfunktion von allen Prozessen geteilt wird. Hinzuzufügen ist der Speicherbedarf aller Programme geringer, da sie sich eine Bibliothek teilen.

Außerdem sollte das Betriebssystem virtuellen Speicher unterstützen, wird die Bibliothek nicht beim ersten Laden komplett übertragen, sondern sie wird in den Speicherbereich der Prozesse die sie verwendet eingeblendet. Danach werden nur benötigte Teile der Bibliothek bei Bedarf der Festplatte in den Arbeitsspeicher von der virtuellen Speicherverwaltung geladen.

4.2 Statische Bibliotheken

Im Gegensatz zur dynamischen Bibliothek wird die statische Bibliothek nach dem Kompiliervorgang durch den Binder oder auch linker, in einem eigenen Schritt mit dem ausführbaren Programm verbunden. Der Binder oder Linker sucht Unterprogramme aus den Bibliotheksdateien, für die es im Programm keine Implementierung gibt. Die Unterprogramme werden dann aus den Dateien extrahiert und an das Programm gebunden, das bedeutet der code des Unterprogramms wird an beim Programmcode hinzugefügt und die Aufruf Verweise werden auf die Unterprogramme Adressen gerichtet.

4.3 Quelltext Bibliotheken

Diese Bibliotheken sind einfache Bibliotheken, Quelltext Bibliotheken enthalten Sammlungen von Wertedefinitionen, Deklarationen, Funktionen, Klassen, generische Bestandteile und so weiter.

5. Compiler

Den F95-Compiler gibt es fast für alle Computer, hinweg von Arbeitsplatzrechner bis zu Supercomputern. Hier gibt es mehrere Hersteller, entweder die Computerhersteller wie IBM, HP und Intel. Es gibt aber auch spezialisierte Softwarehersteller wie Absoft, PGI, NAG, lahey und Silverfrost/Salford. Die F77-Compiler beziehungsweise die reinen F77-Compiler werden heutzutage nicht mehr hergestellt, da Fortran 77 fast vollständig im Sprachstandard von Fortran 95 enthalten ist.

Für manche der oben genannten Compiler ist die nichtkommerzielle Nutzung kostenlos, dies gilt für Privatanwender.

Der GNU Compiler Collection (GCC) ist für fast alle Plattformen vorhanden, dieser enthält seit der Version 4.0 ein Fortran 95-Frontend (gfortran). Die Version 8.0 unterstützt

Hierzu werden wir ein Beispiel mit Cygwin, der einen gfortran Compiler besitzt.

addier.f90

```
1   !
2         !  Ein einfaches FORTRAN-Programm
3         !
4         PROGRAM addier
5          INTEGER :: zahl1, zahl2, summe
6          READ *, zahl1, zahl2
7          summe = zahl1 + zahl2
8          PRINT *, summe
9         END
10
```

Abbildung 1: Programmcode für einen addierer.

Gucken wir uns nun zuerst den code in Abb. 1 an. Dieser ist ein Code für einen Addierer von zwei Zahlen. Die Ausrufezeichen in den Zeilen 1-3 sind zum Ausklammern da, da nur ein einfacher Kommentar zum Verständnis existiert. Nun in der Zeile 4 können wir den Namen des Programms erkennen, welches addier lautet.

In der Zeile 5 haben wir nun festgelegt, dass Integer benutzt werden soll, für zahl1, zahl2 und die summe, das bedeutet alles müssen ganze zahlen ergeben da sonst in der Zeile 6 es zu einem Fehler kommen wird, da hier mit Read überprüft wird, ob es sich um ganze zahlen handeln wird.

Bei der Zeile 7 wird dann nur die Summe berechnet, diese existiert aus Zahl1 addiert mit Zahl2. In der Zeile 8 soll dann die Summe ausgeben werden, bevor das Programm beendet wird in der Zeile 9.

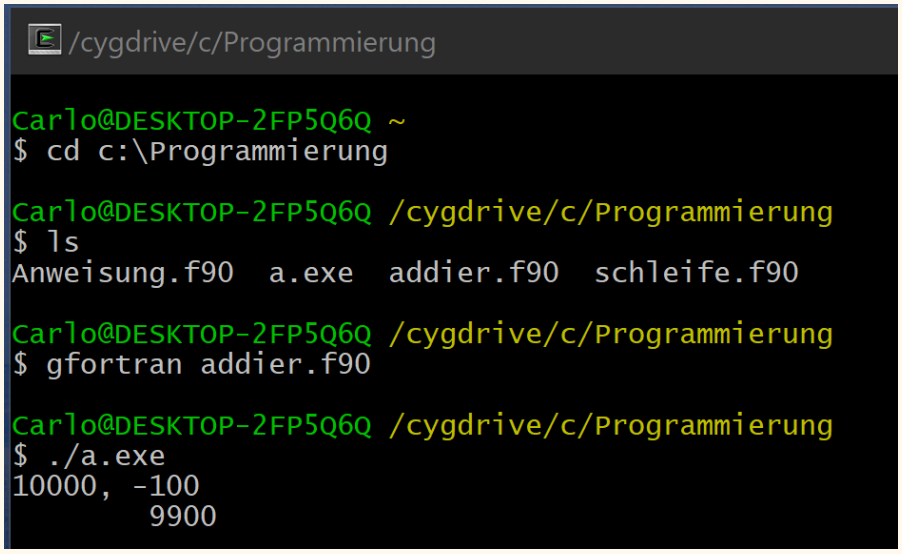

Abbildung 2: Ausführen des Programmcode mit Cygwin

Da Fortran einen Compiler braucht um den Code auch ausführen zu können werden wir dies nun machen. Hierfür verwenden wir Cygwin, der einen gfortran Compiler besitzt.

Zuerst müssen wir in den korrekten Ordner wechseln, dies geschieht durch cd c:\Programmierung. Nun können wir mit ls überprüfen, ob unser Programm sich auch in dem Ordner befindet. Dies tut es, so verwenden wir den gfortran Compiler um das Programm addier.f90 auszuführen. Hier ist es auch wichtig anzumerken, dass beim Name des Programms .f90 steht, dies ist sehr wichtig damit der gfortran das Programm ausführen kann. Ein Programm soll immer die Endung der Version des benutzten gfortran Compiler haben.

Da wir im code keine bestimmten Zahlen festgelegt haben können, wir nun uns welche zahlen aussuchen welche addiert werden sollen und wie von Abb. 2 ablesbar können auch Zahlen im minus Bereich addiert werden, da diese auch noch zu den ganzen

Zahlen gehören. Es ist auch sichtbar, dass das Ergebnis am Ende ausgegeben wird und auch richtig erscheint.

6. Datentypen

Wie bei jeder Programmiersprache gibt es unterschiedliche Datentypen. Bei Fortran gibt es zwei unterschiedliche Datentypen, der Arithmetische und logischer Datentyp.

6.1 Arithmetische Datentypen

Es gibt 4 arithmetische Datentypen, die in Fortran benutzt werden können.

Der erste Datentyp wäre Integer, dieser ist für die ganzen Zahlen zuständig, dies bedeutet damit können die Zahlen wie z. B.: 15, -6500, 200.000.000 dargestellt und zum Rechnen benutzt werden.

Als Nächstes kommt der real Datentyp, dieser ist zuständig für die Gleitkommazahlen einfacher Genauigkeit, damit sind zahlen wie: 3.1415, -5.5, 7e3 gemeint.

Double precision ist ein Datentyp der im Gegensatz zu real die doppelte Genauigkeit der Gleitkommazahlen darstellen und zum Berechnen benutzen kann. Die Zahlen sehen dann ungefähr so aus: 3.1415D0, -5.5D0.

Als Letztes gibt es noch den Datentyp complex, mit diesem Datentypen ist es möglich Komplexe Zahlen (Zwei real-Zahlen) darzustellen und zum Rechnen zu benutzen, hier ein Beispiel wie die Zahlen aussehen können: (3.1415, -5.5), (1.4, 7.1E4).

Bei Fortran die Binär-, Oktal- und Hexadezimalzahlen darzustellen.

Um eine Binärzahl darzustellen, wird der Anfangsbuchstabe geschrieben und darauf folgt die Zahl in Gänsefüßen. Dies sieht so aus B”zahl”, es ist aber auch möglich es so darzustellen b´zahl´. Es gibt also zwei Schreibweisen.

Für die Oktalzahlen, ist das Prinzip das gleiche O am Anfang und die Zahl folgt in Gänsefüßen, also sieht dies so aus O”zahl” ,auch hierbei ist es möglich die Oktalzahlen so O´zahl´ zu schreiben.

Die Hexadezimalzahl folgt einem ähnlichen Prinzip, hier wird jedoch nicht der Anfangsbuchstabe benutzt, sondern das Z von Hexade**z**imalzahl. Das bedeutet die erste Schreibweise sieht wie folgt aus: Z”zahl”. Die zweite Schreibweise sieht dementsprechend so aus: Z´zahl´.

6.2 Logischer Datentyp

Nun kommen wir zum logischen Datentyp.

Es gibt nur einen logischen Datentyp, dieser ist der Logical Datentyp, er ist zuständig für wahr oder falsch. Das bedeutet, das Ergebnis ist entweder true oder false.

6.3 Zeichenketten

Da Fortran für numerische Berechnung vorgesehen ist, haben Zeichenketten eine Art Sonderstellung. Zeichenketten mussten später hinzugefügt werden, da der Markt nicht nur an Berechnungen interessiert war, sondern Programme bedienen wollte.

In Fortran werden Zeichenketten mit Leerzeichen aufgefüllt.

Um eine Zeichenkette zu deklarieren, geschieht dies wie eine normale Variable, der einzige Unterschied ist das kein Datentyp angegeben wird, sondern nach dem Charakter wird n in Klammern angegeben, das n steht für die Zeichenkette mit einer Länge von n Zeichen, sieht dementsprechend so aus: character(n).

Es gibt auch die Möglichkeit ein Stern zwischen dem n und dem Charakter zu schreiben. Dies könnte so aussehen: CHARACTER*8 a, hier würde eine Zeichenkette der Länge 8 Zeichen definiert werden. Das ist nur leider der FORTRAN 77 Stil, welcher nicht mehr verwendet werden sollte.

Um die Länge einer Zeichenkette zu bestimmen wird entweder LEN oder LEN_TRIM benutzt. LEN gibt die Anzahl an vereinbarten Zeichen zurück, während LEN_TRIM die Länge der Zeichenkette zurückgibt.

7. Variablen

Eine Variable ist charakterisiert durch einen symbolischen Namen, einen Datentyp, seinen Wert und dem Speicherplatz.

Beim Programmstart hat eine Variable keine definierten Wert. Es existieren zwei Arten, damit eine Variable ihren Datentyp enthält, einmal durch die implizite Typ Anweisung und die explizite Typ Anweisung.

7.1 Implizite Typ Anweisung

Bei der impliziten Typ Anweisung bestimmt der Anfangsbuchstabe des Variablenbezeichners den Datentyp. Dies bedeutet, wenn im Code implicit angegeben ist, wird die Implizite Typ Anweisung benutzt.

Dies bedeutet es würde bei integer wegen des Anfangsbuchstaben, die Buchstaben I bis N benutzt werden. Bei real würden die restlichen Buchstaben benutzt werden.

Wie im oberen genannten Text gibt es die Anweisung implicit none, die ist dafür zuständig, dass die implizite Typ Anweisung komplett ausgeschaltet wird. Dies liegt daran, dass durch die implizite Typ Anweisung Fehlermöglichkeiten entstehen.

7.2 Explizite Typ Anweisung

Explizite Typ Anweisung ist das Gegenteil von der impliziten Typ Anweisung. Hier bekommt die Variablen den Datentyp von Programmierer zugeordnet. Hier werden die arithmetischen Datentypen und der logische Datentyp benutzt.

Hierzu werd ich ein Beispiel auf der folgenden Seite ausführen.

schleife.f90

```
 1 program schleife
 2 implicit none
 3
 4 integer :: i
 5
 6 do i =1,10
 7
 8 write (*,*)i
 9
10 end do
11 end program schleife
```

Abbildung 3: Programmcode für eine Schleife

In Abb. 3 können Sie den code für eine Schleife erkennen. Wie Sie sehen befindet sich in der ersten Zeile der Name des Programmcodes. In der zweiten Zeile befindet sich das implicit none, welches dafür zuständig ist die implizite Typ Anweisung auszuschalten. In der 4 Zeile erfolgt dann nun die Zuweisung des Datentypen an der Variable, hier soll der Datentyp Integer verwendet werden, dies bedeutet es werden nur ganze zahlen benutzt für diese Schleife.

In der Zeile 6 wird nun angegeben welche der ganzen Zahlen verwendet werden sollen, hierfür werden die zahlen nach der Do-schleife mit einem Komma unterschieden. Darauf folgt das in der achten Zeile diese Nummern aufgeschrieben werden sollen.

Als Letztes soll nun die Do-Schleife beendet werden sollen, bevor das Programm komplett endet.

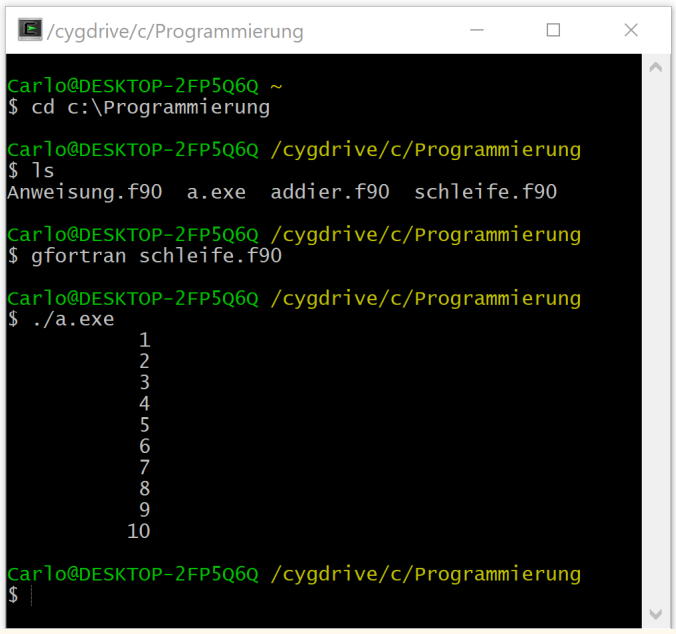

Abbildung 4: Ausführen mit Cygwin Compiler

Wir im vorherigen Beispiel werden wir auch den Gfortran Compiler, in Cygwin benutzen.

Wie in Abb.4 zusehen ist müssen wir wieder in den zuständigen Ordner wechseln, wir tun dies mit cd c:\Programmierung. Wir kontrollieren nun mit ls, ob sich unser Programm schleife.f90 auch im Ordner befindet. Die Abbildung 4 zeigt das dies der Fall ist, nun können wir das Programm mit den gfortran Compiler ausführen.

Im nächsten schritt können wir sehen, dass die Schleife ohne Probleme ausgeführt wird. Wir müssen uns aber auch daran erinnern, dass die Schleife nur angezeigt wird, wegen der Zeile 8 im Code. Sonst wäre die Schleife ausgeführt worden, jedoch nicht aufgeschrieben was zu Verwirrungen führen kann. Es ist aber in der Abbildung zusehen, dass der Code ohne Probleme funktioniert.

8. Literaturverzeichnis

Internet:

-Entwickler. Fortran gestern, heute und morgen
Fortran gestern, heute und morgen - entwickler.de

-Fortran, "Fortran" Links
Deutsche Fortran WebSite

-Proggen, Zeichenketten
proggen.org - fortran:string - Raum für Ideen

-Universität kiel, Fortran: eine ausführliche Minimal-Anleitung
fortran.pdf (uni-kiel.de

-Universität Leipzig, Wissenschaftliches Programmieren in Fortran
Wissenschaftliches Programmieren in Fortran (uni-leipzig.de)

-Wikibooks, Fortran: Einleitung.
Fortran: Einleitung – Wikibooks, Sammlung freier Lehr-, Sach- und Fachbücher

-Wikibooks, Fortran: FORTRAN 77: Verzweigungen und Schleifen
Fortran: FORTRAN 77: Verzweigungen und Schleifen – Wikibooks, Sammlung freier Lehr-, Sach- und Fachbücher

-Wikibooks, Fortran: Fortran 95: Unterprogramme
Fortran: Fortran 95: Unterprogramme – Wikibooks, Sammlung freier Lehr-, Sach- und Fachbücher

-Wikipedia, Typisierung(informatik)
Typisierung (Informatik) – Wikipedia