Carlotta Williams Lopez 7286477

Fortran(***FOR***mula ***TRAN***slation = Formel Übersetzung)

Gliederung

- Generelle Information
- Eigenschaften
- Geschichte
- Bibliotheken
- Vor- und Nachteile
- Zusammenfassung
- Literatur

(Quelle fortran.pdf (uni-kiel.de))

Allgemein

Allgemeiner Aufbau Ein Fortran-Programm besteht aus direkt lesbarem Text, aus Buchstaben (zwischen Groß- und Kleinschreibung unterscheidet der compiler nicht!), Zahlen und einigen wenigen Sonderzeichen. Leerzeichen haben für Fortran-Compiler ¨ keinerlei Bedeutung; im Prinzip können sie sogar innerhalb von statement-Namen oder Variablennamen in beliebiger Zahl auftauchen, aber damit verwirrt man sich nur selber. Im Allgemeinen steht in jeder einzelnen Programmzeile genau ein Befehl (Anweisung, command, statement). Diese Befehle werden bei Ausführung des Programms von oben ¨ nach unten nacheinander abgearbeitet (sofern nicht bestimmte Kontrollstrukturen diesen Ablauf modifizieren). Bei einigen dieser Befehle ” passiert tats¨achlich etwas“ (es wird gerechnet, input oder output erzeugt, usw.); dies sind ” ausführbare Befehle“. Andere Befehle dienen dazu, das ¨ Programm für den compiler zu strukturieren, Variablennamen zu vereinbaren, usw.; sie ¨ sind nicht im eigentlichen Sinne ausführbar. ¨ Das kleinstmögliche Fortran-Programm hat deshalb mindestens zwei Zeilen, tut aber trotzdem gar nichts: program minimal end An der ersten Zeile erkennt der compiler den Beginn eines (Haupt-)Programms, an der letzten Zeile das Ende eines Programmabschnitts (Hauptprogramm oder Unterprogramm). 5Prof. Dr. B. Hartke, Universität Kiel, hartke@pctc.uni-kiel.de Beim program-statement steht ein Name des folgenden Programmabschnitts; dieser Name dient zur Orientierung des Programmierers. Der legale Aufbau eines nicht-trivialen Fortran-Programms sieht so aus: Anfang: program-, subroutine- oder function-statement Deklarationen: Vereinbarung von Variablen u.¨a. Ausführbarer Teil: ¨ alle ausführbaren Anweisungen ¨ Ende: end-statement Insbesondere dürfen Variablendeklarationen und ¨ ¨ähnliche nicht-ausführbare Anweisungen ¨ nicht im ausführbaren Teil vorkommen und umgekehrt. Es gibt keine speziellen Grenzmarken ¨ zwischen diesen Abschnitten; z.B. beginnt der ausführbare Teil schlicht mit der ersten ¨ ausführbaren Anweisung.

Carlotta Williams Lopez 7286477

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

Fortran ist eine Programmiersprache, die insbesondere für numerische Berechnungen eingesetzt wird. Der Name entstand aus *FORmula TRANslation* und wurde bis zur Version FORTRAN 77 mit Großbuchstaben geschrieben.

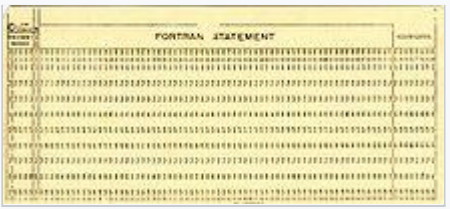
Generell

- Fortran ist eine **höhere Programmiersprache** (Higher Level Language, HLL, Programmiersprache der 3. Generation)
- Fortran ist eine **prozedurale Programmiersprache** (Prozedurale Programmierung)
- Fortran ist eine **imperative Programmiersprache** (Imperative_Programmierung)
- Fortran ist eine **objektorientierte Programmiersprache** (ab Fortran 2003, Objektorientierte Programmierung).
- Fortran verwendet das Konzept der **starken Typisierung**. Fortran kennt **explizite und implizite Typisierung** (Typisierung (Informatik))
- Fortran ist eine sehr alte Programmiersprache, die aber laufend weiterentwickelt und somit den modernen Trends immer wieder angepasst wurde und wird.

Eigenschaften

- Fortran war und ist für numerische Berechnungen vorgesehen und optimiert. Von Anfang an hatte Fortran den Potenz-Operator **. Dieser ist in vielen anderen Hochsprachen nicht vorhanden. Weiters kennt Fortran einen Datentyp für komplexe Zahlen. Mit Fortran 90 wurden Vektor- und Matrix-Operationen standardisiert. Insbesondere für wissenschaftliche und numerische Berechnungen gibt es in FORTRAN umfangreiche Bibliotheken, die immer noch weit verbreitet sind, auch wenn eine zunehmende Menge an Routinen inzwischen nach C und C++ portiert wurde.

Geschichte



Eine FORTRAN-Lochkarte aus den Anfangstagen des Computerzeitalters

Fortran gilt als die erste jemals tatsächlich realisierte höhere Programmiersprache.

Sie geht zurück auf einen Vorschlag, den John W. Backus, Programmierer bei IBM, 1953 seinen Vorgesetzten unterbreitete.

Dem Entwurf der Sprache folgte die Entwicklung eines Compilers durch ein IBM-Team unter Leitung von John W. Backus. Das Projekt begann 1954 und war ursprünglich auf sechs Monate ausgelegt. Tatsächlich konnte Harlan Herrick, der Erfinder der später heftig kritisierten Goto-Anweisung, am 20. September 1954 das erste Fortran-Programm

ausführen. Doch erst 1957 wurde der Compiler für marktreif befunden und mit jedem IBM 704-System ausgeliefert. Backus hatte darauf bestanden, den Compiler von Anfang an mit der Fähigkeit zu Optimierungen auszustatten: er sah voraus, dass sich Fortran nur dann durchsetzen würde, wenn ähnliche Ausführungsgeschwindigkeiten wie mit bisherigen Assembler-Programmen erzielt würden.

Versionen

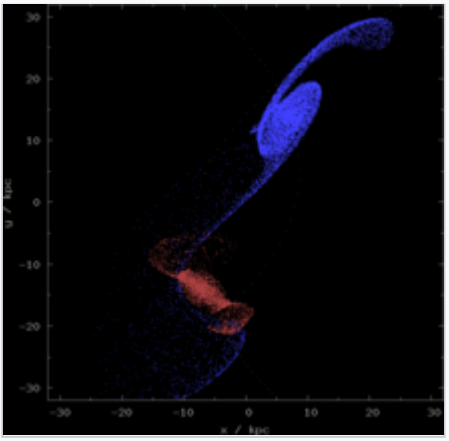
Genealogie

Fortran wurde mehrmals erweitert. Viele neue Sprachelemente wurden zunächst von einem einzelnen Hersteller eingeführt und später in den internationalen Standard übernommen. Als Versionen folgten aufeinander FORTRAN I, FORTRAN II, FORTRAN IV, FORTRAN 66, FORTRAN 77, Fortran 90, Fortran 95, Fortran 2003, Fortran 2008 etc. Ab FORTRAN 66 ist Fortran von einer internationalen Organisation standardisiert. Die Fortschreibung der Standards ist ein komplizierter Prozess, der oft wesentlich länger dauert als zunächst angestrebt: Der Nachfolger des 1978 erschienenen Standards FORTRAN-77, der als Fortran 8x bezeichnet wurde, war ursprünglich für das Jahr 1982 geplant, später dann für das Jahr 1985, und wurde schließlich unter der Bezeichnung Fortran90 erst am 11. April 1991 als neuer Standard und Nachfolger von FORTRAN-77 angenommen.[1]

Im Laufe dieser Erweiterungen wurden zahlreiche Sprachelemente aus neueren Programmiersprachen übernommen. Beruhte früher Fortran-Stil noch ganz auf Goto-Anweisungen, kann man seit FORTRAN 77 uneingeschränkt strukturiert programmieren. Mit Fortran 90 wurde das aus der Lochkartenzeit stammende Zeilenformat freigegeben. Ab Fortran 90 wurden interessante Elemente eingeführt, die auch z.B. in Ada vorhanden sind, beispielsweise optionale Parameter und die Möglichkeit, Prozedurparameter nicht nur über die Position in der Parameterliste zu identifizieren, sondern über ihren Namen.

Ein Überblick über die historische Entwicklung von Fortran

Carlotta Williams Lopez 7286477



Simulation von Galaxien mittels Fortran-Programm

Jahr	Version	Anmerkungen
1954 bis 1957	FORTRAN, FORTRAN I	entwickelt von einem IBM-Team unter Leitung von John W. Backus, war dies die erste wirklich erfolgreiche höhere Programmiersprache
1957/58	FORTRAN II	Inline-Assembler, Kommentare, u.a.
1958	FORTRAN III	einige kleinere Änderungen, wurde aber nie offiziell freigegeben
1961/62	FORTRAN IV	eine verbesserte und erweiterte Version
1966	FORTRAN 66	die erste standardisierte Fortran-Version und gleichzeitig überhaupt die erste standardisierte höhere Programmiersprache

Carlotta Williams Lopez 7286477

1978	FORTRAN 77	DO-Schleife, IF THEN-ELSE IF-ELSE-END IF, CHARACTER-Datentyp, u.a.
1991	Fortran 90	free form style, Module, Zeiger, Datenverbund, u.v.m.
1997	Fortran 95	kleinere Änderungen
2003	Fortran 2003	OOP, C-Binding, u.a.
2010	Fortran 2008	Co-Arrays und kleinere Änderungen
2018	Fortran 2018	kleinere Änderungen
?	Fortran 202x	

Varianten

Einige von Fortran abgeleitete Programmiersprachen bzw. Dialekte von Fortran sind beispielsweise Ratfor, F und HPF (High Performance Fortran). Auf Fortran aufgesetzt ist das Finite-Elemente-Programmpaket Nastran.

Einordnung von Fortran

-

Popularität

Die Popularität einer Programmiersprache einigermaßen fundiert zu bestimmen ist nicht einfach. Dennoch gibt es Institutionen, die das versuchen, sei es über die Anzahl von Einträgen in Suchmaschinen, Zugriffstatistiken für Internetseiten, Nutzerbefragungen oder

auch zeitliche Veränderungen bei Buchverkäufen. Hier wird stellvertretend eine dieser Statistiken angeführt:

Lt. TIOBE lag Fortran im April 2021 hinsichtlich Popularität mit einem Rating von 0,91% an 20. Stelle von insgesamt 100 gelisteten Programmiersprachen. Das ist zwar weit hinter dem führenden C oder dem zweitplatzierten Java. Dennoch rangiert Fortran in dieser Statistik vor anderen bekannten Programmiersprachen, wie z.B. Haskell oder Smalltalk (TPCI - TIOBE Programming Community Index).

Solche Statistiken sind natürlich mit Vorsicht zu geniessen, aber in Form eines groben Richtwerts "Daumen mal Pi" können sie schon einen ersten Eindruck von Verbreitung und Popularität einer Programmiersprache geben.

Im Bereich der numerische Datenverarbeitung, insbesondere auf Hochleistungsrechnern, ist Fortran gemeinsam mit C/C++ nach wie vor führend.

Quelltext Bibliotheken

Quelltextbibliotheken enthalten Sammlungen von Wertedefinitionen, Deklarationen, Funktionen, Klassen, generischen Bestandteilen, usw.

API[Bearbeiten]

Eine Programmierschnittstelle ist eine Schnittstelle die von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird. Oft wird dafür die Abkürzung *API* (für engl. *application programming interface*, deutsch: *Schnittstelle zur Anwendungsprogrammierung*) verwendet. Im Gegensatz zu einer Binärschnittstelle (ABI) definiert ein API nur die Verwendung der Schnittstellen auf Quelltextebene.

Neben dem Zugriff auf Datenbanken, die Hardware wie Festplatte oder Grafikkarte kann ein API auch das Erstellen von Komponenten der grafischen Benutzeroberfläche ermöglichen oder vereinfachen.

Im weiteren Sinne wird die Schnittstelle jeder Bibliothek (Library) als API bezeichnet.

Statische Bibliotheken

Statische Bibliotheken werden nach dem Kompiliervorgang durch einen so genannten Linker oder Binder in einem eigenen Schritt mit dem ausführbaren Programm verbunden.

Der Linker sucht aus den Bibliotheksdateien Unterprogramme heraus, für die es im Programm keine Implementierung gibt. Diese werden dann aus den Dateien extrahiert und an das Programm gebunden, d.h. der Unterprogrammcode wird an den Programmcode angefügt und die Aufrufverweise werden auf die Unterprogrammadressen gerichtet.

Dynamische Bibliotheken

Dynamische Bibliotheken werden erst bei Bedarf in den Arbeitsspeicher geladen und durch den sogenannten Lader mit dem ausführbaren Programm verbunden. Dadurch muss eine

Carlotta Williams Lopez 7286477

Bibliothek, die von mehreren Programmen genutzt wird, nur einmal im Speicher gehalten werden.

Dies ist beispielsweise bei Multitasking-Systemen vorteilhaft, wenn die Bibliotheken insgesamt sehr groß sind und von vielen Prozessen gleichzeitig verwendet werden. Dort wird eine Bibliotheksdatei bei ihrer ersten Verwendung in den Speicher geladen. Trifft ein Programm auf den Verweis zu einem Unterprogramm, das noch nicht eingebunden wurde, dann wird ein Laufzeitbinder aktiviert. Dieser sucht das Unterprogramm in den im Speicher vorhandenen Bibliotheken, fügt die Adresse am Aufrufpunkt ein und führt das Unterprogramm erstmalig aus.

Bei jedem weiteren Aufruf des Unterprogramms ist dann die Adresse vorhanden, so dass das Unterprogramm direkt aufgerufen wird. Die Ausführungszeit, insbesondere die Startzeit eines Programms, ist hier geringfügig erhöht. Dies wird in Kauf genommen, da der Programmcode der Bibliotheksfunktionen von allen Prozessen geteilt wird. Der Speicherbedarf aller Programme zusammen ist daher in der Regel kleiner als beim statischen Linken.

Unterstützt das Betriebssystem virtuellen Speicher, so entfällt das Laden der gesamten Bibliothek bei der ersten Verwendung. Stattdessen wird die Bibliothek in den Speicherbereich jedes sie verwendenden Prozesses eingeblendet. Die virtuelle Speicherverwaltung lädt danach nur tatsächlich benötigte Teile der Bibliothek bei Bedarf von der Festplatte in den Arbeitsspeicher.

Bibliotheken in verschiedenen Programmiersprachen

Bibliotheken in Programmiersprachen enthalten Leistungen, die nicht im Compiler implementiert sind, sondern in der Sprache selbst programmiert sind und mit dem Compiler zusammen oder völlig von ihm getrennt dem Programmierer zur Verfügung stehen. Im ersten Fall ist die Bibliothek meist in der Sprachbeschreibung festgelegt. Im zweiten Fall spricht man von einer externen Bibliothek.

Bibliotheken bei verschiedenen Betriebssystemen

Windows[Bearbeiten]

Bei den Betriebssystemen Windows und auch bei OS/2 wird eine Bibliotheksdatei, die dynamisch bindet, als Dynamic Link Library (DLL) bezeichnet. Entsprechend haben diese Dateien meist die Dateiendung *.dll*. Ihr Dateiformat ist *Portable Executable*.

Problematisch ist bei Windows 95, Windows 98 und Windows Me, dass durch unzureichende Schutzmaßnahmen die DLLs nicht kontrolliert werden - jedes Programm darf sie austauschen und kann dem Betriebssystem damit möglicherweise Schaden zufügen. Windows 2000 und Windows XP hingegen verfügen über einen Systemschutz, der auch die DLLs einbezieht.

Vorteile

- Außer Code können auch Daten (z. B. Dialog-Ressourcen) von mehreren Prozessen gemeinsam genutzt werden.
- DLLs werden häufig statisch gelinkt, können aber auch dynamisch (daher der Name) gelinkt werden. Dynamisch heißt hier, dass die DLL explizit vom

Programm zur Laufzeit geladen wird und die Funktionen, die sich in der DLL befinden, „per Hand“ mit dem Programm verbunden werden. Dadurch wird es möglich, durch Austauschen der DLL die Funktionalität des Programms zur Laufzeit zu verändern.

- DLLs können unabhängig vom Hauptprogramm gewartet werden. D. h. Funktionen in der DLL können ohne Wissen des Programms verändert werden. Danach wird die DLL einfach ausgetauscht (die alte DLL-Datei wird überschrieben), ohne dass das Hauptprogramm verändert werden muss.
- Da die DLL als unabhängige Datei dem Hauptprogramm beiliegen muss, können Anbieter von Programmcode besser sicherstellen, dass Programmierer, die die Funktionen ihrer DLL nutzen, dafür auch bezahlen. Die Funktionalität der DLL verschwindet so nicht (wie bei einer Library) im Code des Programms. Dieser Vorteil wird von Befürwortern freier Software als Nachteil gesehen.

Nachteile

Änderungen in DLLs ziehen oft auch Änderungen im Programm mit sich. Dadurch kommt es leicht zu Versionskonflikten, die oft nur sehr schwer aufzuspüren sind.

Eine der Grundideen der DLLs war, Programmcode zwischen mehreren Programmen zu teilen, um so kostbaren Speicher zu sparen. In der Praxis ist es jedoch dazu gekommen, dass viele Programme bei der Installation DLLs in das Windows-Systemverzeichnis schreiben, die außer diesem speziellen Programm kein anderes benutzen kann.

Außerdem ist die Entwicklung und insbesondere die Anbindung im Vergleich aufwändiger als zur statischen Bibliothek.

Quintessenz

DLLs sollte man nur benutzen, wenn man ihre spezielle Funktionalität benötigt und man ausschließlich unter Windows arbeitet. Sind statische Bibliotheken für den Zweck ausreichend, sollte man diese vorziehen. In der Praxis ergeben sich keinerlei Einsparungen bei der Größe des Codes.

Unix-artige

Auf Unix-artigen Betriebssystemen ist für dynamische Bibliotheken die Bezeichnung *shared library* (englisch *shared*, geteilt) gebräuchlich.

Für diese Dateien hat sich die Endung .so (***s**hared **o**bject*) eingebürgert. In der Regel folgt dem Bibliotheksnamen noch eine Versionsnummer.

Carlotta Williams Lopez 7286477

(Quelle Fortran gestern, heute und morgen - entwickler.de)

Fortran („Formular Translation“) ist eine der ältesten „hochsprachlichen“ Programmiersprachen und wird vor allem für sehr rechenintensive Anwendungen genutzt. Im Laufe der Zeit wurde der Sprachstandard immer weiter entwickelt und die Oberflächen wurden den modernen Gewohnheiten angepasst. Die aktuellen Compiler stehen ihren Verwandten aus anderen Sprachwelten in nichts nach. Grund genug für einen Rückblick auf vergangene Zeiten und einen Überblick über die aktuellen Fortran Compiler auf dem deutschen Markt.

„Fortran? Wer programmiert denn heute noch in Fortran?“ Diese Frage hören wir, seit es unser Unternehmen gibt, also seit 1989, fast täglich. Und die Antwort auf diese Frage hat sich seitdem auch nicht wesentlich geändert. Fortran ist weiterhin eine weit genutzte Programmiersprache und wird vor allem dort eingesetzt, wo es wirklich viel zu rechnen gibt. Klassische Anwendungsbereiche gibt es z.B. in der Physik oder den Ingenieurswissenschaften, wo umfangreiche und komplexe mathematische Berechnungen durchgeführt werden. Sehr hilfreich sind hier die sehr guten mathematischen Bibliotheken, die es für diverse Fortran Compiler gibt.

(Quelle Wissenschaftliches Programmieren in Fortran (uni-leipzig.de))

Warum Fortran?
-Felder (Vektoren, Matrizen, ...) sind Teil der Sprache hochoptimierend (Schleifen, Konzept reiner Funktionen, keine Aliasing-Probleme) ⇒ sehr schneller Code
-Sauberer Umgang mit Fließkommazahlen,
-saubere Optimierungen Zahlreiche Bibliotheken, insbesondere zur Numerik (Lapack,...)
-herstellerunabhängige Standardisierung
-Höchstleistungsrechnen, große numerische Probleme: “number crunching”
-Wird in der Industrie nach wie vor eingesetzt
-Flache Lernkurve

Warum nicht Fortran?
-hardwarenahe Programme, Web Applikationen, Datenbanken, ego shooter games,...