KERNFORSCHUNGSZENTRUM

KARLSRUHE

Mai 1973 KFK 1669

Institut für Neutronenphysik und Reaktortechnik

# Eine Methode zur Realisierung dynamischer Strukturen in IBM-FORTRAN

G. Buckel, W. Höbel

GESELLSCHAFT
FÜR
KERNFORSCHUNG M.B.H.

KARLSRUHE

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.
KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

KFK 1669

Institut für Neutronenphysik und Reaktortechnik

# Eine Methode zur Realisierung dynamischer Strukturen in IBM - FORTRAN

G. Buckel
W. Höbel

Gesellschaft für Kernforschung m.b.H., Karlsruhe

Zusammenfassung:

In diesem Bericht wird eine Methode zur Realisierung von dynamischen Strukturen in IBM-Fortran-Programmen beschrieben. Als Hilfsmittel werden die Listen notwendiger Hilfsprogramme sowie ein Demonstrationsbeispiel beigefügt.

A Method for Dynamic Structuring in IBM-FORTRAN.

Abstract:

A method is described in this report to realise dynamical structures in IBM-Fortran-programmes. The listings of all necessary auxiliary-programmes are given and also an example for demonstration.

Inhalt :

## 0. Einleitung

Die in diesem Bericht zugrunde liegende Terminologie beruht auf dem IBM-Sprachgebrauch. Da die hier verwendeten Fachausdrücke nicht alle ausführlich erklärt werden können, sollte in Zweifelsfällen auf die einschlägige IBM-Literatur zurückgegriffen werden. Eine geeignete Auswahl ist unter /1/, /2/, /3/, /4/ und /5/ angegeben.

Dynamische Strukturen stellen eine wichtige Erweiterung der gewöhnlichen Überlagerungsstrukturen (planned overlays) dar. Auf die Unterschiede wird im nächsten Abschnitt eingegangen. Zunächst soll jedoch der Rahmen für die Anwendbarkeit der Methode abgesteckt werden.

Die Umsetzung technisch-wissenschaftlicher Aufgaben in Rechenprogramme geschieht nach wie vor sehr häufig mit Hilfe von Fortran. Diese Programmiersprache hat sich als leicht erlernbar und vielfältig anwendbar erwiesen, so daß häufig einige Nachteile in Kauf genommen werden. Die in diesem Bericht vorgeschlagene Einführung von dynamischen Strukturen in IBM-Fortran-Programmen kann zur Behebung zweier solcher Mängel auf IBM-Rechenanlagen dadurch führen, daß

a) die Anpassung von Fortranprogrammen an die gegebene DV-Umgebung erleichtert und

b) eine sehr flexible Verknüpfung von Großprogrammen ermöglicht wird.

Unter gegebener "DV-Umgebung" ist dabei die Gesamtheit aller Hilfsmittel eines Rechners, wie Kernspeicher, Rechen- und Steuerwerke, externe Speicher usw. sowie die heterogene Menge aller um diese Hilfsmittel konkurrierenden Programme - und damit alle Benutzer der Anlage - zu verstehen. Anpassung bedeutet in diesem Zusammenhang für ein Programm die Beachtung der durch die o.e. Hilfsmittel vorgegebenen Grenzen, wie Speichergröße, Anzahl der Datenträger usw. Diese physikalischen Grenzen können durch die Betriebssysteme kurzfristig verändert werden. So sind im Falle von Multiprogrammierung (im Folgenden kurz: MVT = multiprogramming with a variable number of tasks) die Hilfsmittel unter mehrere gleichzeitige Bewerber möglichst optimal aufzuteilen. In einer MVT-Umgebung ist es deshalb zweckmäßig und u.U. sogar notwendig, daß sich große Programme (Größenordnung: $10^3$ bis einige $10^4$ Fortran-Anweisungen) durch eine effektive Überlagerungsstruktur verschiedenen Kernspeichergrößen anpassen können. Dies gilt ganz speziell für die zentrale Rechnerkombination IBM/360-65 und /370-165 des Kernforschungszentrums Karlsruhe.

Die unter b) angesprochene "flexible Programmverknüpfung" bezieht sich nicht so sehr auf die in einem Fortranprogramm normalerweise angewandte Unterprogrammtechnik als vielmehr auf die Verknüpfung von

Zum Druck eingereicht am 17.4.1973

Großprogrammen zur Behandlung von "dynamischen Aufgabenstellungen". Solche Aufgabenstellungen treten bei der numerischen Behandlung von reaktorphysikalischen Problemen, etwa bei der orts- und zeitabhängigen Neutronenflußberechnung oder bei der Berechnung des Langzeitverhaltens eines Reaktors auf. Das "Dynamische" in der Problematik besteht darin, daß die zu durchlaufende Programmfolge von Zwischenresultaten abhängt und u. U. nicht vorausgesagt werden kann, so daß planned-overlay-Strukturen zu unübersichtlich und uneffektiv werden und als Hilfsmittel ausscheiden. Dies gilt insbesondere für den Fall rekursiver Prozeduren.

Die weiter unten beschriebene Einführung von dynamischen Strukturen in IBM-Fortran ist das Ergebnis einer Versuchsreihe, die beim Aufbau des Karlsruher nuklearen Programmsystems KAPROS (vgl. /6/) mit dem Ziel durchgeführt wurde, ein geeignetes Programmanagement für dieses modulare System zu finden. Auch bei anderen modularen Programmsystemen, z. B. bei dem amerikanischen ARC-System (vgl. /7/), wurden dynamische Strukturen in ähnlicher Form für Fortran implementiert.

## 1. Die Ausführung von Fortran-Programmen unter dem IBM-Betriebssystem

Gegenstand dieses Abschnitts sind die Programmstrukturen, die das IBM-Betriebssystem zur Ausführung von Fortran-Programmen zuläßt. Zuvor wird jedoch in erlaubter Kürze auf die Objekte des Programmanagements sowie auf die Konventionen zur Programmverknüpfung eingegangen. Wenn dabei zunächst von vereinfachten Vorstellungen ausgegangen wird, bedeutet dies dennoch keine Einschränkung für die Anwendung der in diesem Bericht dargestellten Methode.

Normalerweise werden alle in Frage kommenden Fortran-Programme mit Hilfe der Fortran-Compiler in eine maschinennahe, Assembler-ähnliche Form - in sog. 'Objekt-Programme' - umgewandelt und solange aufbewahrt, bis sie in einem zweiten Schritt durch den Linkage Editor (vgl. /4/) zu einem ausführbaren Programm, einem sog. Lademodul (oft auch kurz: Modul), zusammengefaßt worden sind. Dabei werden i. a. mehrere Objektprogramme samt aller notwendigen Dienstprogramme wie System-Ausgabe, Sinus-Cosinus-Funktionen, etc. zu einem einzigen Lademodul zusammengefaßt. Diesem Lademodul wird zum Zwecke kurz- oder langfristiger Speicherung bzw. des Wiederauffindens ein möglichst eindeutiger Name (später auch: Modulname), zugeordnet. Danach kann er unter Angabe dieses Namens in etwaigen Kontrollanweisungen oder Makro-Instruktionen, die teilweise weiter unten angeführt werden, in der Regel direkt zur Ausführung kommen.

Wird bei einem Programmablauf in einem Lademodul die serielle Befehlsfolge durch den Aufruf eines anderen Programmsegments (Control Section, siehe /3/) bzw. eines anderen Unterprogramms unterbrochen, so gelten für diese Verbindungsstellen spezielle Regeln, die eingangs erwähnten Konventionen zur Programmverknüpfung (linkage conventions in /3/). Diese Regeln verlangen die Aufbewahrung der alten Registerinhalte (zum Zeitpunkt des Absprungs) in vorgegebenen, von den Compilern bereitgestellten Speicherbereichen - den sog. Save Areas - innerhalb des rufenden Programms und legen die Verwendung einzelner Register zur Programmverknüpfung fest (Einsprungadresse im allgemeinen Register 15, Rücksprungadresse im a.R. 14, Save-Area-Adresse des rufenden Programms im a.R. 13, Adresse der Parameterliste im a.R. 1, etc.). Anhand der Save-Area-Inhalte, die auch untereinander Verweise enthalten, kann der Programmablauf jederzeit verfolgt und kontrolliert werden - auch visuell im Falle eines Fehlerabbruchs mittels eines Kernspeicherauszugs (Dump).

Ein einfacher Sprung von einem Programm in ein Unterprogramm kann von vornherein ohne Mithilfe des Betriebssystems ausgeführt werden, wenn der zugehörige Lademodul im Kernspeicher geladen werden kann. Ist jedoch letzteres nicht der Fall, weil beispielsweise sehr viele oder sehr große Unterprogramme vorhanden sind, so können über die eingangs erwähnten Programmstrukturen Hilfsmittel des Betriebssystems bereitsgestellt werden, die dennoch eine Ausführung des Lademoduls gestatten. Das IBM-Betriebssystem bietet zur Herstellung von Programmstrukturen in Lademoduln durch den Linkage Editor drei Möglichkeiten:

- einfache Struktur (Simple Structure)
- Überlagerungsstruktur (Planned Overlay)
- dynamische Struktur (Dynamical Structure)

Die Merkmale dieser Strukturen, die - mit Ausnahme der letzten - auch für Fortran-Programme realisierbar sind, werden anschließend erläutert.

## 1.1. Die einfache Struktur

Die einfache Struktur besteht aus einem einzigen Lademodul, der alle erforderlichen Befehle, Unterprogramme und Datenfelder enthält. Er wird als ganzes in den Kernspeicher gebracht und verbleibt dort für die gesamte Ausführungsdauer.

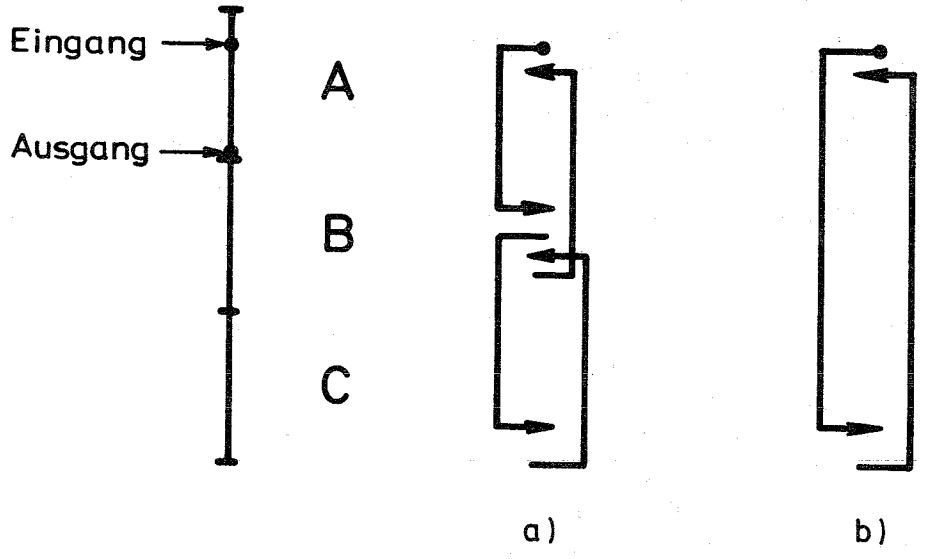


Fig. 1 : Darstellung einer einfachen Programmstruktur

In der Fig1 sind eine aus den drei Programmen A, B, C bestehende einfache Struktur sowie einige mögliche Programmdurchläufe dargestellt. Vom Haupt- oder Steuerprogramm A, das vom Betriebssystem gestartet wird, geht die Kontrolle weiter an B und C, C kehrt zurück nach B und B nach A (Fall a)) oder A ruft C, Ckehrt unmittelbar nach A zurück (Fall b)). Die Verzweigungen erfolgen ohne Eingriffe des Betriebssystems.

Der Vorteil dieser Struktur besteht offensichtlich in der permanenten Verfügbarkeit aller auszuführenden Befehlsfolgen im Kernspeicher, wodurch zeitraubende und kostspielige Ein- und Ausgabevorgänge zu deren Bereitstellung überflüssig sind. Dagegen kann die relativ starre Struktur großer Programme bei der Speicherung hinderlich sein. Außerdem ist die Verknüpfung mehrerer Strukturen dieser Art zu einer einzigen Programmausführung (Job-Step-Task) nicht möglich. Des weiteren stellt die Belastung des Kernspeichers mit nicht weiter benötigten Programmteilen eine Verschwendung teuerer Hilfsmittel dar. Darüberhinaus wird der Vorteil des Einsparens von Ein- Ausgabe-Aktionen in einer MVT-Umgebung nahezu hinfällig, da während der Wartezeit andere Benutzer bedient werden können.

Der Aufbau der einfachen Strukturen erfolgt allein auf der Ebene des Betriebssystems (Job Control Language, Linkage-Editor-Eingabe). Zu ihrer Ausführung sind auf Fortran-Ebene keinerlei Maßnahmen notwendig.

## 1.2 Überlagerungsstrukturen

Bei der Überlagerungsstruktur werden ebenfalls alle notwendigen Unterprogramme in einem einzigen Lademodul zusammengefaßt. Im Unterschied zur einfachen Struktur ist eine Gruppierung der einzelnen Programme zu Programmsegmenten (z.B. E+F in Fig. 2) und der Programmsegmente zu Programmzweigen (A+B+D in Fig. 2) in der Form einer Baumstruktur realisierbar, wie dies in Fig. 2 dargestellt wird.

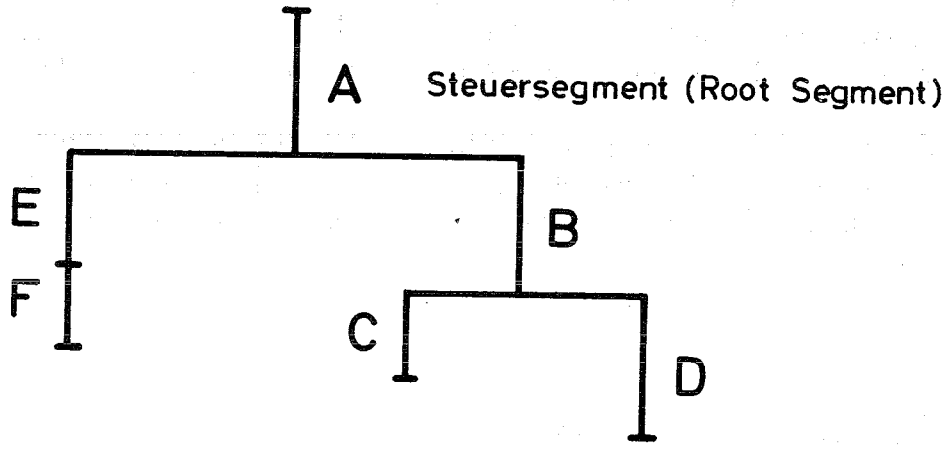


Fig. 2: Einfaches Beispiel einer Überlagerungsstruktur

Anhand der Baumstruktur wird der Kernspeicher nach und nach durch die aufgerufenen Segmente so beladen, daß lückenlose Pfade in den einzelnen Zweigen entstehen. Dabei werden i.a. Segmente anderer Zweige überladen. Am Beispiel der Fig. 2 bedeutet dies, daß beim Übergang der Kontrolle von Segment A zu Segment B sowohl A als auch B geladen sind, beim Übergang von A unmittelbar nach D aber der ganze rückwärtsgerichtete Zweig A+B+C. Sprünge von Zweig zu Zweig, d.h. direkt von D nach C oder von C nach F sind unzulässig (vgl. /4/). Wird jedoch ein neuer Zweig auf reguläre Weise aufgerufen, etwa durch die Aufruffolge A, D-A, F, so erfolgt eine Überlagerung des vorhergehenden, d.h. bei obiger Aufruffolge überlagert das Segment EF die Segmente B+D.

Die Flexibilität von Überlagerungsstrukturen bei der Realisierung von Aufrufen ist recht groß, wenn sie auch durch die vorgegebene Baumstruktur eingeschränkt wird. So sind auch die Lademoduln vieler Fortran-Großprogramme in dieser Struktur gehalten, wobei die durch die Eingriffe des Betriebssystems verursachte Effektivitätseinbuße zufolge notwendiger Ladevorgänge der Zweige in MVT-Umgebungen wiederum gering ist.

Große Hindernisse bestehen für sehr umfangreiche Programmsysteme dagegen in der starren Baumstruktur und in der Beschränkung auf einen einzigen Lademodul, erstere bei der Behandlung "dynamischer Aufgabenstellungen", wenn während des Programmablaufs neue Zweige aufzubauen wären, und letztere wegen der von einem Lademodul nicht überschreitbaren Maximalgröße von 524248 Bytes. Diese maximale Modulgröße ist deshalb kritisch, weil in verschiedenen Zweigen benutzte Exemplare ein- und desselben Programms unter verschiedenen Namen mehrfach vorhanden sein müssen, wodurch Überlagerungsstrukturen über Gebühr aufgeblasen werden können. Hinzu tritt als weiterer Nachteil die Tatsache, daß der Kernspeicherbedarf einer Überlagerungsstruktur durch ihren längsten Zweig bestimmt wird, unabhängig davon, ob dieser Zweig zur Ausführung gelangt oder nicht.

Als Vorteil kann gewertet werden, daß der Aufbau einer Überlagerungsstruktur allein auf Betriebssystemebene durchgeführt wird, und daß zu ihrer Ausführung auf Fortran-Sprachebene keine spezifischen Angaben erforderlich sind.

## 1.3. Dynamische Strukturen

Dynamische Strukturen bieten gegenüber den bisher betriebenen Strukturen die Möglichkeit, mehrere Lademoduln - die selbst wiederum einfache Struktur, Überlagerungsstruktur oder dynamische Struktur haben können, während eines Programmablaufs zu verketten. Um welche Lademoduln es sich bei dieser Verkettung handelt, wird erst zur Zeit der Ausführung bestimmt - und zwar durch die Lademoduln selbst, man kann somit von einer "dynamischen Aufruffolge" von Lademoduln sprechen.

Die Steuerung des Programmablaufs, d.h. der Aufruffolge, beginnt und endet in dem ersten angelaufenen Lademodul, der hinfort "Steuermodul" genannt wird. Sie kann Fortsetzungen in den zugeladenen Lademoduln haben. Da in Fortran entsprechende Sprachelemente zum Aufruf anderer Lademoduln *a priori* nicht vorhanden sind, müssen die entsprechenden Hilfsmittel des Betriebssystems durch Fortran-verträgliche, in Assembler erstellte Hilfsprogramme bereitgestellt werden. Bei den Hilfsmitteln handelt es sich um

Macro-Instruktionen wie LOAD, LINK, ATTACH etc., die das Aufsuchen von Lademodulen in Hierarchien von Programmbibliotheken, das Laden von deren Kopien zwecks Ausführung und die Verzweigung der Ablaufkontrolle in CALL-RETURN-ähnlicher Art (normaler Unterprogrammaufruf) zwischen geladenen Lademoduln ermöglichen.

Die drei zuletzt aufgeführten Funktionen kann das LINK-macro, dessen Anwendung im folgenden Abschnitt näher erläutert wird, durchführen. Weiterführende Information über Programmverkettung in dynamischen Strukturen sind in /3/ bzw. /5/ angegeben. Hier ist noch zu bemerken, daß erstens die Anwendung von LINK Eingriffe des Betriebssystems bewirken, die bei starker Häufung zu spürbaren Rechenzeiteinbußen führen können, und daß zweitens - anders als bei Überlagerungsstrukturen - nur die verketteten Lademoduln den Kernspeicher belegen, und dies nur solange, wie die Verkettung besteht.

Die Flexibilität dynamischer Strukturen in Bezug auf realisierbare Programmabläufe ist nahezu unbeschränkt. Selbst rekursive Prozeduren sind auf diese Weise mit Lademoduln im Rahmen der gegebenen Kernspeichergröße ausführbar. So sind diese Strukturen gut geeignet zum Aufbau großer Programmsysteme mit umfangreichen Programmbibliotheken wie KAPROS/6/ oder ARC /7/. In KAPROS ist darüberhinaus eine Auslagerung der rufenden Moduln nach dem Kellerungsprinzip eingeführt, wodurch der Speicherbedarf drastisch reduziert werden kann. Beim Aufbau von Programmbibliotheken kommt zustatten, daß - wiederum anders als bei Überlagerungsstrukturen - eine Mehrfachspeicherung eines an mehreren Stellen einer Aufruffolge benutzten Lademoduls vermeidbar ist. Allerdings müssen beim Aufsuchen der Lademoduln in allen möglichen Bibliotheken kostspielige Wartezeiten und Ein-Ausgabe-Vorgänge so weit als möglich vermieden werden.

## 2. Realisierung und Hilfsprogramme

Die Realisierung der im 1. Abschnitt beschriebenen Vorstellungen über dynamische Aufruffolgen von Programmen in IBM-FORTRAN wurde folgendermaßen versucht:

Es war beabsichtigt, dynamisch aus einer Benutzerbibliothek Moduln mit Hilfe des Makros LINK /5/, von einem Steuermodul oder von einem bereits zugeladenen Modul aus, in den Kernspeicher zu laden und anzusteuern. Mit dem Makro-LINK können aber in dieser Weise nur fertige Lademoduln behandelt werden, d.h. Moduln, bei denen alle nach der Übersetzung der einzelnen Unterprogramme noch offenen Verknüpfungen durch den Linkage Editor /4/ hergestellt wurden.

Dies bedeutet aber, daß der Linkage Editor im Normalfall die bei der Übersetzung der Ein-Ausgabe Befehle im FORTRAN-Programm offen gebliebenen Referenzen löst, indem er das Paket der Ein-Ausgabe-Hilfsprogramme - kurz IBCOM# genannt - aus der Automatic Call Library / 4 / hinzufügt.

Dies führt dann aber dazu, daß in der dynamischen Aufruffolge jeder einzelne Lademodul mit einem eigenen Programmpaket IBCOM# arbeitet. Das wiederum bedeutet, daß von jedem Lademodul für dieselbe Ein-Ausgabe-Einheit eigene Pufferspeicher angelegt und auch wieder freigegeben werden. Dies ist nicht nur aus Gründen der Kernspeicherausnutzung unwirtschaftlich, sondern auch das IBM Betriebssystem (Operating System OS) /8/ kann bei dieser Behandlung die Übersicht über angelegte, gelöschte und noch bestehende Pufferspeicher verlieren, was letztlich zum Verlust von Ein-Ausgabe-Information führen kann.

Um zu vermeiden, daß die in dynamischer Aufruffolge zu behandelnden Lademoduln einschneidende Konventionen für das Anlegen und Freigeben von Pufferspeichern unterworfen werden müssen, was letzten Endes einer Reduktion des Sprachenumfangs von FORTRAN gleichgekommen wäre, wurden Überlegungen in anderer Richtung angestellt.

Es entstand die Idee, alle Ein-Ausgabe-Operationen über ein einziges, zentral und stationär im Steuermodul liegendes Exemplar des Programms IBCOM# abzuwickeln. Dies setzt dann voraus, daß alle Lademoduln der entsprechenden Benutzerbibliothek ein Exemplar eines eigens anzufertigenden Unterprogramms, im folgenden "Pseudo IBCOM# " genannt, enthalten. Dieses Programm Pseudo-IBCOM# hat dann beim Aufruf als Folge der Ausführung einer Ein-Ausgabe-Operation im Lademodul lediglich einen Sprung an die richtige Stelle des zentral im Lademodul liegenden Programms IBCOM# auszuführen.

Damit diese Vorstellung verwirklicht werden kann, muß dieses Programm Pseudo-IBCOM# ganz bestimmte Voraussetzungen erfüllen:

1.) Es muß formal die Bezeichnung IBCOM# haben. Unter dieser Bezeichnung wird das Paket der Ein-Ausgabe-Programme im übersetzten FORTRAN-Programm aufgerufen.

2.) Es muß im Primary Input für den Linkage Editor angeliefert werden. Bei zwei Programmen gleichen Namens zieht der Linkage Editor das im Primary Input dem aus der Automatic Call Library vor.

3.) Das Paket der echten Ein-Ausgabe-Programme IBCOM# besteht im wesentlichen aus einem Steuerteil und den eigentlichen Verarbeitungsprogrammen. Vom FORTRAN Programm aus, das Ein-Ausgabe Operationen durchführt, erfolgt der Absprung in den Steuerteil, wobei die einzelnen Funktionen (READ, WRITE, REWIND usw.) nur durch die Einsprungstelle im Steuerteil unterschieden werden. Die Einsprungstellen legt der FØRTRAN Compiler durch verschiedene Inkremente fest, bezogen auf den Programmanfang von IBCOM# Von diesen Einsprungstellen aus werden lediglich Programmsprünge in die jeweiligen Verarbeitungsprogramme ausgeführt. Das Programm Pseudo-IBCOM# muß dann die selben Einsprungstellen simulieren und von dort aus die Sprünge an die richtigen Stellen im echten IBCOM# durchführen.

4.) Damit dies durchgeführt werden kann, muß Pseudo-IBCOM# einen Programmteil besitzen, der die Anfangsadresse des echten IBCOM# als Argument beim Aufruf übernimmt und in Pseudo IBCOM# überträgt.

5.) Damit formal nicht nur Unterprogramme, sondem auch Hauptprogramme in dynamischer Aufruffolge behandelt werden können, muß der Einsprung in das echte Programm IBCOM# , der den Programmstop behandelt, abgefangen werden. Im Pseudo-IBCOM# wird dazu ein Sprung in einen Programmteil am Ende ausgeführt, der den Rücksprung in das rufende Programm durch das Laden der Register aus der zuständigen Savearea / 3 / und durch Nullsetzen des Fehlercodes in Register 15 vorbereitet und anschließend ausführt.

6.) Im Prolog von Hauptprogrammen wird grundsätzlich die Standard-Eingabe-Einheit durch einen speziellen Einsprung in das Programm IBCOM# eröffnet. Für sämtliche zugeladenen Moduln wurde dieser Vorgang bereits im Steuermodul erledigt. Der entsprechende Einsprung in das Programm Pseudo-IBCOM# enthält deshalb lediglich einen Sprungbefehl zurück in den rufenden, zugeladenen Modul.

7.) Damit der Lademodul von der jeweils speziellen Anordnung der einzelnen Unterprogramme beim Verknüpfungsvorgang im Linkage-Editor unabhängig wird, kann das Programm Pseudo-IBCOM# einen weiteren Programmteil enthalten. Dieser führt dann lediglich einen Sprung an den Programmanfang des Moduls aus. Der Name dieses Programmteils wird dann dem Linkage Editor als Ansprungspunkt (Entry Point) für den betreffenden Lademodul eingegeben.

Das Programm Pseudo-IBCOM# auf Seite 16 erfüllt diese 7 Voraussetzungen. Die in den Punkten 4.) und 7.) genannten Programmteile haben die Einsprungspunkte IBCO und ABC. Aus dem oben gesagten ergibt sich zwingend, daß jeder Lademodul als erste ausführbare Anweisung den Programmteil IBCO aufzurufen hat. Die Adresse des echten, zentral im Steuermodul gelegenen Programms IBCOM# ist dann das einzige zu übertragende Argument.

Ein Steuermodul verschafft sich die Adresse des echten Programms IBCOM# als Argument beim Aufruf des Hilfsprogramms IBCENT. Ruft der Steuermodul einen weiteren Modul auf, so liefert er ihm die Adresse des echten IBCOM# in der Argumentliste an. Der Modul selbst gibt diese Adresse als Argument beim Aufruf des Programmteils IBCO an sein Exemplar von Peudo-IBCOM# weiter. Ruft der gerufene Modul einen weiteren Modul, so gibt er diesem die Adresse des echten IBCOM# als Argument weiter.

Der Aufruf eines Lademoduls selbst geschieht durch das Linkprogramm LINK auf Seite 15, das im wesentlichen das Makro LINK aufruft. Dabei ist die Übertragung von Argumentlisten möglich. Das Programm LINK wurde im wesentlichen vom Programmsystem ARC aus Argonne übernommen / 9 /. Die

Benutzerbibliothek, welche die zuzuladenden Moduln enthält, muß dem Betriebssystem (OS) durch eine Daten-Definitions-Karte als Job- oder Steplib-Karte / 8 / bekanntgemacht werden.

Bei der Aufnahme der zuzuladenden Moduln in die Benutzerbibliothek sind die Modulnamen dem Linkage Editor unter dem NAME-Parameter einzugeben, unter denen sie später aufgerufen werden sollen.

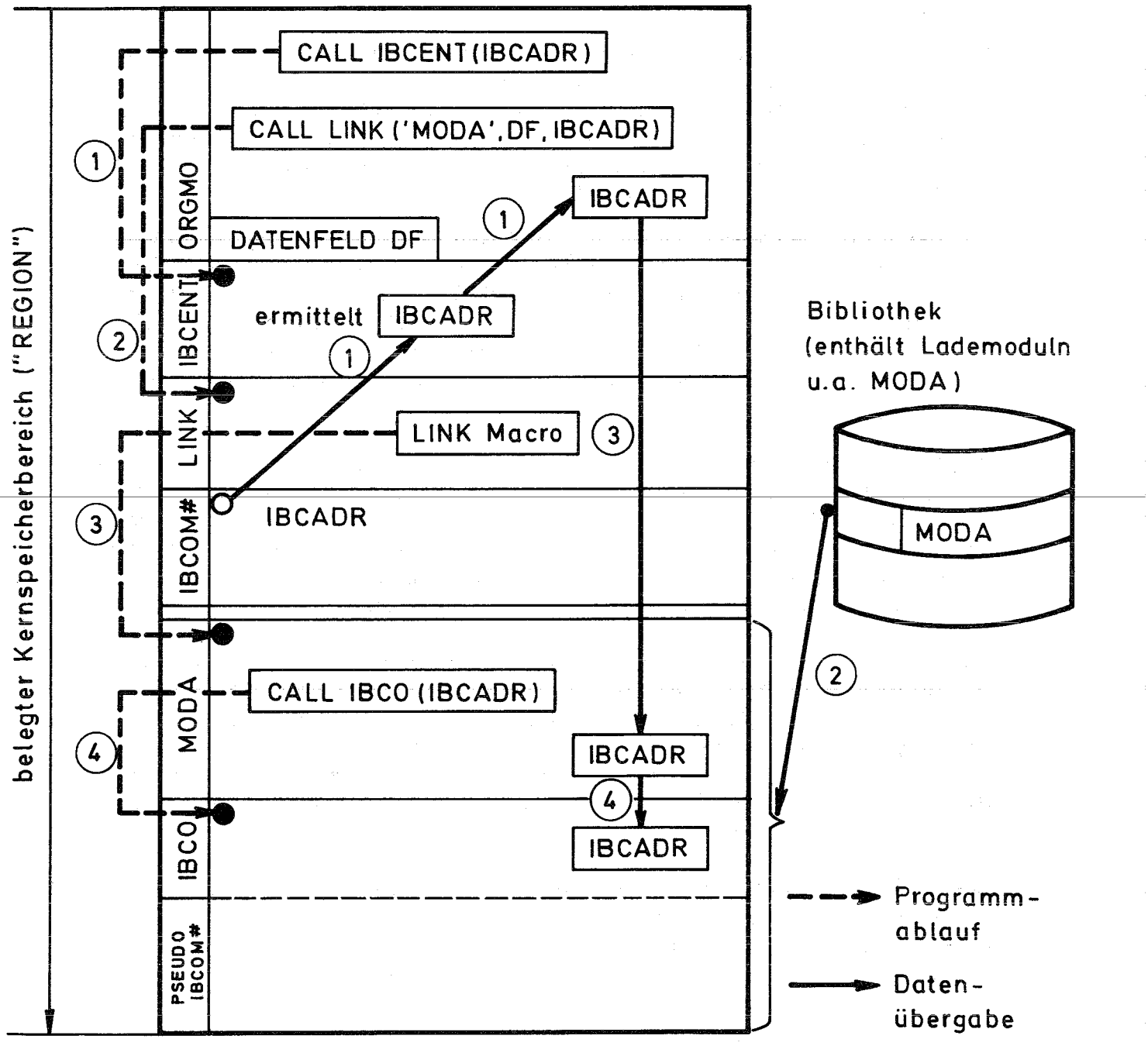


Fig. 3: Aufruf des Lademoduls MODA durch den Steuermodul ØRGMØ

Figur 3 soll den gesamten Programmablauf beim Aufruf eines Lademoduls MODA von einem Steuermodul ØRGMØ (Organisationsmodul) aus, veranschaulichen.

Die Hilfsprogramme LINK, IBCENT und ABC sind in einem "partitioned data set" mit der Bezeichnung OBJ.NUSYS gespeichert und können mit der auf Seite 17 angegebenen Folge von Job Control Karten dem Linkage Editor zur Bildung des Steuermoduls zugänglich gemacht werden.

## 3. Beispiel für einen dynamischen Modulaufruf mit Erläuterung.

Der in Fig. 4 schematisch dargestellte Modulaufruf soll dynamisch verwirklicht werden, indem die Namen der zuzuladenden und aufzurufenden Moduln vom Benutzer frei gewählt und über äußere Eingabe dem Steuermodul zugänglich gemacht werden.

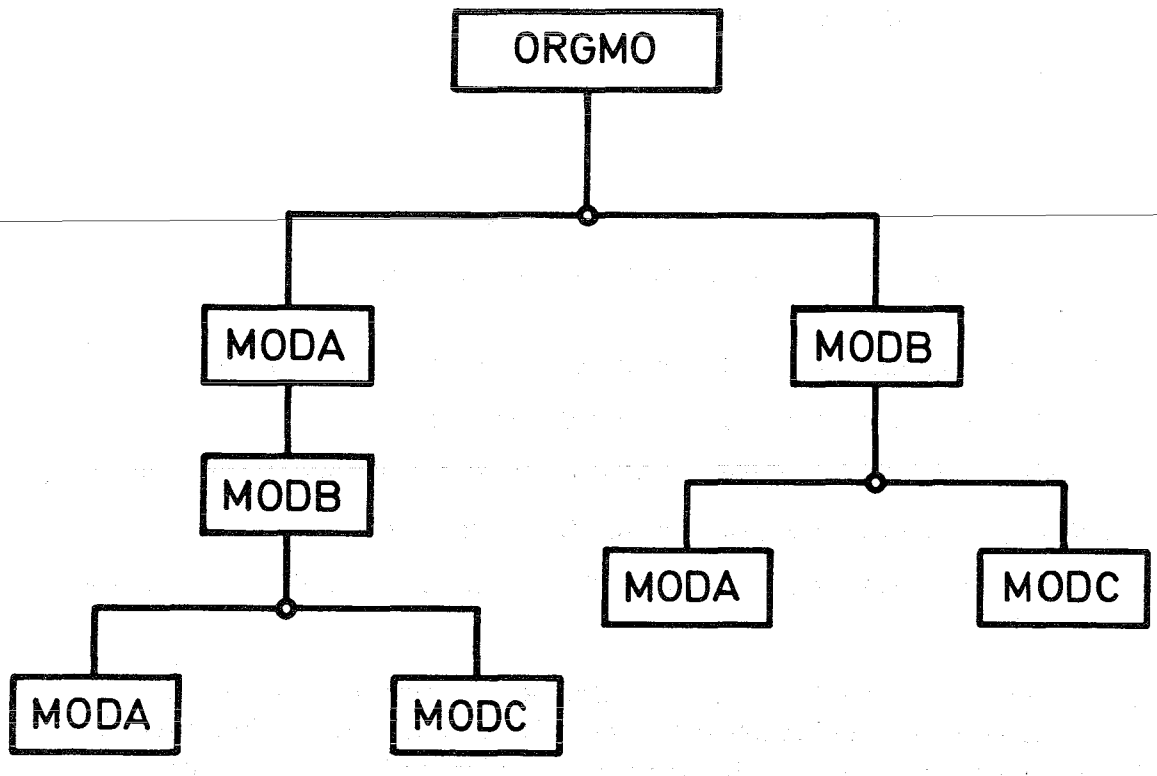


Fig. 4 : Beispiel einer dynamisch realisierbaren Modulfolge.

Im Steuermodul ORGMO (Organisationsmodul) wird vom Standard-Eingabefile der Name des zuzuladenden Moduls $\left\{\begin{matrix}\text{MODA}\\\text{MODB}\end{matrix}\right\}$ gelesen. Außerdem liest ORGMO den Namen des später von MODB wahlweise zuzuladenden Moduls $\left\{\begin{matrix}\text{MODA}\\\text{MODC}\end{matrix}\right\}$ und schreibt diesen Namen auf einen Zwischenfile, dessen Nummer NZWF ebenfalls vom Standard-Eingabefile gelesen wird.

Im Zweig 1 wird zunächst der Modul MODA zugeladen, der seinerseits den Modul MODB aufruft. MODB liest den Namen des weiter zuzuladenden Moduls $\left\{\begin{matrix}\text{MODA}\\\text{MODC}\end{matrix}\right\}$ vom Zwischenfile NZWF und ruft den entsprechenden Modul auf.

Im Zweig 2 wird zunächst der Modul MODB zugeladen, der vom Zwischenfile NZWF den Namen des weiter zuzuladenden Moduls $\left\{\begin{matrix}\text{MODA}\\\text{MODC}\end{matrix}\right\}$ liest und diesen aufruft.
Auf Seite18 ist die Fortran-Liste des Steuermoduls ORGMO abgebildet. In der Variablen IEND wird der Endindex für eine Schleife eingelesen. Diese Variable ist im betrachteten Beispiel =5 gesetzt, d.h., daß der in Figur 4 schematisch dargestellte Modulablauf 5 mal durchgeführt wird. Die zugehörige Eingabe der Namen zuzuladender Moduln und der Nummern der Zwischenfiles lautet im betrachteten Beispiel:

| | | | |
|---|---|---|---|
| 1.) | MODA | MODC | 15 |
| 2.) | MODB | MODA | 14 |
| 3.) | MODA | MODA | 15 |
| 4.) | MODB | MODC | 14 |
| 5.) | MODA | MODC | 15 |

Dies hatte im Einzelnen folgende Modulabläufe zur Folge:

1.) ORGMO → MODA → MODB → MODC → MODB → MODA → ORGMO

2.) ORGMO → MODB → MODA → MODB → ORGMO

3.) ORGMO → MODA → MODB → MODA → MODB → MODA → ORGMO

4.) ORGMO → MODB → MODC → MODB → ORGMO

5.) ORGMO → MODA → MODB → MODC → MODB → MODA → ORGMO

Auf den Seiten 19, 20 sind die Fortran-Listen der Lademoduln MODA, MODB und MODC abgebildet. Sie bestehen im wesentlichen aus einem Schreibbefehl auf die Standard-Ausgabe-Einheit, der den ordnungsgemäßen Aufruf des betreffenden Lademoduls dokumentieren soll.

Für praktische Beispiele kann man etwa an verschiedene Integrations- oder Interpolations-Moduln denken, deren Aufruf der geforderten Genauigkeit in der gerechneten Aufgabe entsprechend gewählt werden kann.

Die Seiten 21, 22 enthalten die Originalausgaben der einzelnen Moduln auf die Standard-Ausgabe-Einheit, welche die oben genannte Aufruffolge bestätigt.

Literaturhinweise:

/ 1 / IBM System / 360 Operating System
Fortran G und H Programmers Guide Form C 28-6817

/ 2 / IBM System / 360 Operating System
Concepts and Facilities Form C 28-6535

/ 3 / IBM System / 360 Operating System
Supervisor and Data Management Services Form C 28-6646

/ 4 / IBM System / 360 Operating System
Linkage Editor Form C 28-6538

/ 5 / IBM System / 360 Operating System
Supervisor and Data Management Macro Instructions Form C 28-6647

/ 6 / H. Bachmann e.a.: Das modulare Programmsystem KAPROS
KFK in Vorbereitung

/ 7 / B.F. Toppel: The Argonne Reactor Computation System (ARC)
ANL 7332

/ 8 / IBM System / 360 Operating System
Job Control Language Form C 28 - 6539

/ 9 / L. Just : private Mitteilung 1968

```
 1 *BEIM AUFRUF VON  I B C E N T  WIRD DIE EINGANGSADRESSE IN DIE
 2 *FORTRAN-I/O-ROUTINE IBCOM# FESTGESTELLT UND ZURUECKGEGEBEN.
 3 *AUFRUF:   CALL IBCENT(K)
 4 *  K = EINGANGSADRESSE VON IBCOM# BEIM RUECKSPRUNG.
 5 *
 6 IBCENT  START 0
 7         BC    15,12(15)
 8         DC    C'IBCENT  '
 9         STM   14,3,12(13)
10         BALR  3,0
11         USING *,3
12         L     2,0(1)
13         L     1,IBCAD
14         ST    1,0(2)
15 RETURN  LM    14,3,12(13)
16         MVI   12(13),X'FF'
17         BC    15,0(14)

18 IBCAD   DC    V(IBCOM#)
19         END   IBCENT


 1 *      PROGRAMM ZUM RUFEN EINES LOADMODULS BEI DYNAMISCHER STRUKTUR
 2 LINK     START
 3          SAVE  (14,12),,*
 4+         B     10(0,15) BRANCH AROUND ID
 5+         DC    AL1(4)
 6+         DC    CL4'LINK' IDENTIFIER

 7+         STM   14,12,12(13) SAVE REGISTERS
 8          BALR  2,0
 9          USING *,2
10          ST    13,SAVEBLK+4
11          LR    12,13
12          LA    13,SAVEBLK
13          ST    13,8(12)
14          L     3,0(1)
15          MVC   LNKMC+12(8),0(3)
16          A     1,=F'4'
17 LNKMC    LINK  EP=MODULE
18+LNKMC    DS    0H
19+         CNOP  0,4
20+         BAL   15,*+20 LOAD SUP.PARAMLIST ADR
21+         DC    A(*+8) ADDR OF EP PARAMETER
22+         DC    A(0) DCB ADDRESS PARAMETER        LCOA
23+         DC    CL8'MODULE' EP PARAMETER
24+         SVC   6 ISSUE LINK SVC
25          LTR   15,15
26          BNZ   ABEND
27 RETURN   L     13,4(13)
28          RETURN (14,12)
29+         LM    14,12,12(13) RESTORE THE REGISTERS
30+         BR    14 RETURN
31 ABEND    ABEND 13,DUMP
32+ABEND    DS    0H
33+         CNOP  0,4
34+         B     *+8 BRANCH AROUND CONSTANT
35+         DC    AL1(128) DUMP/STEP CODE
36+         DC    AL3(13) COMPLETION CODE
37+         L     1,*-4 LOAD CODES INTO REG 1
38+         SVC   13 LINK TO ABEND ROUTINE

39 SAVEBLK  DC    18F'0'
40          END   LINK
41                =F'4'
```

```
 1 *         PROGRAMM PSEUDO-IBCOM# ZUR ERMOEGLICHUNG DYNAMISCHER
 2 *         AUFRUFFOLGEN IN FORTRANPROGRAMMEN.
 3 *         EINSPRUNGSTELLE FUER DEN JEWEILIGEN LADEMODUL.
 4 ABC       CSECT
 5           ENTRY     IBCOM#
 6           L         15,8(15)
 7           BR        15

 8 VCON      DC        V(MAIN)
 9 *         HAUPTTEIL VON PSEUDO-IBCOM#  -  SIMULATION DER ECHTEN
10 *         IBCOM#-EINSPRUENGE.
11 IBCOM#    DS        0F
12           BC        15,72(15)  OPEN READ, FORMATGEBUNDEN
13           BC        15,80(15)  OPEN WRITE, FORMATGEBUNDEN
14           BC        15,88(15)  OPEN, FORMATGEBUNDEN MIT VARIABLENLISTE
15           BC        15,96(15)  OPEN, FORMATGEBUNDEN MIT FELDLISTE
16           BC        15,104(15) CLOSE, FORMATGEBUNDEN
17           BC        15,112(15) OPEN READ, FORMATFREI
18           BC        15,120(15) OPEN WRITE, FORMATFREI
19           BC        15,128(15) OPEN, FORMATFREI MIT VARIABLENLISTE
20           BC        15,136(15) OPEN, FORMATFREI MIT FELDLISTE
21           BC        15,144(15) CLOSE, FORMATFREI
22           BC        15,152(15) BACKSPACE
23           BC        15,160(15) REWIND
24           BC        15,168(15) ENDE DES FILES
25 *         BEHANDLUNG DES PROGRAMMSTOPS BEI HAUPTPROGRAMMEN
26 *         DURCH SPRUNG IN DEN ENTSPRECHENDEN PROGRAMMTEIL.
27           BC        15,240(15) STOP UND STOP(I)
28           BC        15,184(15) PAUSE
29           BC        15,192(15) EXEC FEHLER-MONITOR
30 *         BEHANDLUNG DES OPEN FUER DIE STANDARD-EINGABE-EINHEIT
31 *         IM PROLOG VON HAUPTPROGRAMMEN DURCH EINFACHEN
32 *         RUECKSPRUNG.
33           BC        15,0(14)   INTERRUPT-PROCESSOR
34           BC        15,204(15) JOB-TERMINATOR
35           L         15,236(15)
36           BC        15,0(15)
37           L         15,236(15)
38           BC        15,4(15)
39           L         15,236(15)
40           BC        15,8(15)
41           L         15,236(15)
42           BC        15,12(15)
43           L         15,236(15)
44           BC        15,16(15)
45           L         15,236(15)
46           BC        15,20(15)
47           L         15,236(15)
48           BC        15,24(15)
49           L         15,236(15)
50           BC        15,28(15)
51           L         15,236(15)
52           BC        15,32(15)
53           L         15,236(15)
54           BC        15,36(15)
55           L         15,236(15)
56           BC        15,40(15)
57           L         15,236(15)
58           BC        15,44(15)
59           L         15,236(15)
```

```
60            BC          15,48(15)
61            L           15,236(15)
62            BC          15,52(15)
63            L           15,236(15)
64            BC          15,56(15)
65            L           15,236(15)
66            BC          15,60(15)
67            BC          15,64(15)
68            L           15,236(15)
69            BC          15,68(15)
70 *          PROGRAMMTEIL FUER DIE UEBERTRAGUNG DER ECHTEN
71 *          IBCOM#-ADRESSE.
72            ENTRY       IBCC
73 IBCC       ST          2,28(13)
74            L           2,0(1)
75            L           2,0(2)
76            ST          2,24(15)
77            L           2,28(13)
78            BCR         15,14
79 *          PROGRAMMTEIL ZUR BEHANDLUNG DES PROGRAMMSTOPS
80 *          IM EPILOG VON HAUPTPROGRAMMEN.
81            CNOP        0,4
82            DS          XL4'00000000'
83            L           13,4(13)
84            L           14,12(13)
85            LM          0,12,20(13)
86            L           15,260(15)
87            BC          15,0(14)
88            DC          XL4'00000000'
89            END         IBCOM#
```

---

```
JOB ORIGIN FROM LOCAL DEVICE=RD2      ,02C.
//INR168RT JOB (0168,101,P6M2G),BUCKEL,CLASS=B,REGION=240K
//JOBLIB DD DSN=LOAD.NUSYS,UNIT=3330,DISP=SHR
// EXEC FHCLG,PARM.C='MAP,OPT=2,LIST',TIME=(0,50)
//C.SYSIN DD *
/*
//L.SYSLIN DD
// DD DSNAME=OBJ.NUSYS(LINK),DISP=OLD
// DD DSNAME=OBJ.NUSYS(IBCENT),DISP=OLD
// DD *
/*
//G.FT06F001 DD  DCB=(BLKSIZE=133,RECFM=FA,LRECL=133,BUFNO=1)
//G.SYSUDUMP DD SYSOUT=A
//G.FT14F001 DD UNIT=SYSDA,SPACE=(TRK,(1)),DISP=(NEW,DELETE),
// DCB=(BLKSIZE=400,RECFM=VBS)
//G.FT15F001 DD UNIT=SYSDA,SPACE=(TRK,(1)),DISP=(NEW,DELETE),
// DCB=(BLKSIZE=400,RECFM=VBS)
//G.SYSIN DD *
/*
//
```

```
C                                                                       ORGMO
C                                                                       ORGMO
C     STEUERPROGRAMM ZUR ORGANISATION DER AUFRUFFOLGE DER TESTMODULN    ORGMO
C     MODA, MODB UND MODC.                                              ORGMO
C                                                                       ORGMO
C     BENOETIGT WERDEN DIE HILFSPROGRAMME IBCENT UND LINK.              ORGMO
C                                                                       ORGMO
C     DEFINITION DER VARIABLEN UND FELDER:                              ORGMO
C          1.)  IBCADR: ADRESSE DES ECHTEN PROGRAMMPAKETS DER EIN- AUS= ORGMO
C                       GABE-ROUTINEN IBCOM# IM LADEMODUL              ORGMO
C          2.)  NFE:    NUMMER DER STANDARD - EINGABEEINHEIT           ORGMO
C          3.)  NFA:    NUMMER DER STANDARD - AUSGABEEINHEIT           ORGMO
C          4.)  NZWF:   NUMMER EINER EXTERNEN HILFSEINHEIT             ORGMO
C          5.)  DF,JDF: SPEICHERFELD                                    ORGMO
C          6.)  LDIM:   DIMENSION DES SPEICHERFELDS DF, BZW. JDF        ORGMO
C          7.)  KENNZ:  KENNZEICHEN ZUR STEUERUNG DES MODULABLAUFS      ORGMO
C          8.)  MOD1:   AUFZURUFENDER MODUL                             ORGMO
C          9.)  MOD2:   AUFZURUFENDER MODUL                             ORGMO
C                                                                       ORGMO
      DIMENSION DF(100),JDF(100)                                        ORGMO
      REAL*8 MOD1,MOD2                                                  ORGMO
C                                                                       ORGMO
      EQUIVALENCE (DF(1),JDF(1))                                        ORGMO
C                                                                       ORGMO
C                                                                       ORGMO
      LDIM=100                                                          ORGMO
      NFE=5                                                             ORGMO
      NFA=6                                                             ORGMO
      CALL IBCENT (IBCADR)                                              ORGMO
C                                                                       ORGMO
      WRITE (NFA,5)                                                     ORGMO
    5 FORMAT (1H1,'ERSTE AUSGABE DES STEUERPROGRAMMS.')                 ORGMO
      READ (NFE,8) IENDE                                                ORGMO
    8 FORMAT (I4)                                                       ORGMO
C                                                                       ORGMO
      DO 20 I=1,IENDE                                                   ORGMO
      READ (NFE,10) MOD1,MOD2,NZWF                                      ORGMO
   10 FORMAT (2A8,I4)                                                   ORGMO
      WRITE (NFA,15) MOD1,MOD2,NZWF                                     ORGMO
   15 FORMAT (1H0/1H0/                                                  ORGMO
     1         33H0  AUFRUFFOLGE DER MODULN:  1.)  ,A4/                 ORGMO
     2         33H                            2.)  ,A4/                 ORGMO
     3         1H0,'2. MODUL WURDE AUF FILE ' I4,'  UEBERTRAGEN')       ORGMO
      REWIND NZWF                                                       ORGMO
      WRITE (NZWF) MOD2                                                 ORGMO
      REWIND NZWF                                                       ORGMO
      KENNZ=1                                                           ORGMO
C                                                                       ORGMO
      CALL LINK  (MOD1,LDIM,DF,JDF,NFE,NFA,NZWF,KENNZ,IBCADR)           ORGMO
   20 CONTINUE                                                          ORGMO
      WRITE (NFA,25)                                                    ORGMO
   25 FORMAT (1H0,'PROGRAMM ORDNUNGSGEMAESS BEENDET,'/                  ORGMO
     1     1H ,'NACHDEM ALLE AUFRUFE RICHTIG AUSGEFUEHRT WURDEN.')      ORGMO
      STOP                                                              ORGMO
      END                                                               ORGMO
```

```
C                                                                       MODA
C                                                                       MODA
C     MODA IST EINER VON 3 TESTMODULN, DIE VON EINEM STEUERPROGRAMM ZUR  MODA
C     DEMONSTRATION EINER DYNAMISCHEN AUFRUFFOLGE AUS EINER BIBLIOTHEK  MODA
C     ABGERUFEN WERDEN.                                                 MODA
C                                                                       MODA
C     BENOETIGT WERDEN DIE HILFSPROGRAMME IBCOM# (MIT ENTRY IBCO) UND   MODA
C     LINK.                                                             MODA
C                                                                       MODA
C     DEFINITION DER VARIABLEN UND FELDER:                              MODA
C          1.) IBCADR: ADRESSE DER ECHTEN EIN- AUSGABEROUTINEN IBCOM# IM MODA
C                      STEUERMODUL DIE IN PSEUDOIBCOM# UEBERTRAGEN WIRD. MODA
C          2.) NFE:    STANDARD-EINGABEEINHEIT                          MODA
C          3.) NFA:    STANDARD-AUSGABEEINHEIT                          MODA
C          4.) NZWF:   EXTERNE HILFSEINHEIT                             MODA
C          5.) DF,JDF: SPEICHERFELD                                     MODA
C          6.) LDIM:   DIMENSION VON DF BZW. JDF                        MODA
C          7.) KENZ:   KENNZEICHEN ZUR STEUERUNG DES MODULABLAUFS       MODA
C                                                                       MODA
C                                                                       MODA
      SUBROUTINE MAIN (LDIM,DF,JDF,NFE,NFA,NZWF,KENNZ,IBCADR)           MODA
C                                                                       MODA
C                                                                       MODA
      DIMENSION DF(LDIM),JDF(LDIM)                                      MODA
C                                                                       MODA
      CALL IBCO (IBCADR)                                                MODA
C                                                                       MODA
      WRITE (NFA,5) LDIM                                                MODA
    5 FORMAT (1HO,'DER MODUL  ***  MODA  ***  WURDE RICHTIG ANGELAUFEN'/ MODA
     1      1HO,'DABEI WURDE EIN FELD DER LAENGE',I8,'  UEBERTRAGEN.')  MODA
C                                                                       MODA
      IF (KENNZ) 20,20,10                                               MODA
C                                                                       MODA
   10 CALL LINK ('MODB    ',LDIM,DF,JDF,NFE,NFA,NZWF,KENNZ,IBCADR)      MODA
C                                                                       MODA
      WRITE (NFA,15)                                                    MODA
   15 FORMAT (1HO,'RUECKSPRUNG AUS MODB IN MODA WURDE ORDNUNGSGEMAESS AU MODA
     1SGEFUEHRT.')                                                      MODA
C                                                                       MODA
   20 RETURN                                                            MODA
      END                                                               MODA
```

```
C                                                                       MODB
C                                                                       MODB
C     MODB IST EINER VON 3 TESTMODULN, DIE VON EINEM STEUERPROGRAMM ZUR  MODB
C     DEMONSTRATION EINER DYNAMISCHEN AUFRUFFOLGE AUS EINER BIBLIOTHEK  MODB
C     ABGERUFEN WERDEN.                                                 MODB
C                                                                       MODB
C     BENOETIGT WERDEN DIE HILFSPROGRAMME IBCOM# (MIT ENTRY IBCO) UND   MODB
C     LINK.                                                             MODB
C                                                                       MODB
C     DEFINITION DER VARIABLEN UND FELDER:                              MODB
C          GENAU DIESELBE WIE IN MODA.                                  MODB
C                                                                       MODB
C                                                                       MODB
      SUBROUTINE MAIN (LDIM,DF,JDF,NFE,NFA,NZWF,KENNZ,IBCADR)           MODB
C                                                                       MODB
C                                                                       MODB
      DIMENSION DF(LDIM),JDF(LDIM)                                      MODB
      REAL*8 MODUL                                                      MODB
C                                                                       MODB
      CALL IBCO (IBCADR)                                                MODB
C                                                                       MODB
      WRITE (NFA,5) LDIM                                                MODB
    5 FORMAT (1HO,'DER MODUL  ***  MODB  ***  WURDE RICHTIG ANGELAUFEN'/ MODB
     1        1HO,'DABEI WURDE EIN FELD DER LAENGE',I8,'  UEBERTRAGEN.') MODB
C                                                                       MODB
      READ (NZWF) MODUL                                                 MODB
      KENNZ=0                                                           MODB
C                                                                       MODB
      CALL LINK (MODUL,LDIM,DF,JDF,NFE,NFA,NZWF,KENNZ,IBCADR)           MODB
C                                                                       MODB
      WRITE (NFA,10) MODUL                                              MODB
   10 FORMAT (1HO,'RUECKSPRUNG AUS  ',A4,'  IN MODB WURDE RICHTIG AUSGEF MODB
     1UEHRT.')                                                          MODB
C                                                                       MODB
      RETURN                                                            MODB
      END                                                               MODB


C                                                                       MODC
C                                                                       MODC
C     MODC IST EINER VON 3 TESTMODULN, DIE VON EINEM STEUERPROGRAMM ZUR  MODC
C     DEMONSTRATION EINER DYNAMISCHEN AUFRUFFOLGE AUS EINER BIBLIOTHEK  MODC
C     ABGERUFEN WERDEN.                                                 MODC
C                                                                       MODC
C     BENOETIGT WIRD DAS HILFSPROGRAMM IBCOM# (MIT ENTRY IBCO)          MODC
C                                                                       MODC
C     DEFINITION DER VARIABLEN UND FELDER:                              MODC
C          GENAU DIESELBE WIE IN MODA.                                  MODC
C                                                                       MODC
C                                                                       MODC
      SUBROUTINE MAIN (LDIM,DF,JDF,NFE,NFA,NZWF,KENNZ,IBCADR)           MODC
C                                                                       MODC
C                                                                       MODC
      DIMENSION DF(LDIM),JDF(LDIM)                                      MODC
C                                                                       MODC
      CALL IBCO (IBCADR)                                                MODC
C                                                                       MODC
      WRITE (NFA,5) LDIM                                                MODC
    5 FORMAT (1HO,'DER MODUL  ***  MODC  ***  WURDE RICHTIG ANGELAUFEN'/ MODC
     1        1HO,'DABEI WURDE EIN FELD DER LAENGE',I8,'  UEBERTRAGEN.') MODC
C                                                                       MODC
      RETURN                                                            MODC
      END                                                               MODC
```

ERSTE AUSGABE DES STEUERPROGRAMMS.

```
  AUFRUFFOLGE DER MODULN:  1.)  MODA
                           2.)  MODC
2. MODUL WURDE AUF FILE   15  UEBERTRAGEN
DER MODUL  ***  MODA  ***  WURDE RICHTIG ANGELAUFEN
DABEI WURDE EIN FELD DER LAENGE      100  UEBERTRAGEN.
DER MODUL  ***  MODB  ***  WURDE RICHTIG ANGELAUFEN
DABEI WURDE EIN FELD DER LAENGE      100  UEBERTRAGEN.
DER MODUL  ***  MODC  ***  WURDE RICHTIG ANGELAUFEN
DABEI WURDE EIN FELD DER LAENGE      100  UEBERTRAGEN.
RUECKSPRUNG AUS  MODC  IN MODB WURDE RICHTIG AUSGEFUEHRT.
RUECKSPRUNG AUS MODB IN MODA WURDE ORDNUNGSGEMAESS AUSGEFUEHRT.

  AUFRUFFOLGE DER MODULN:  1.)  MODB
                           2.)  MODA
2. MODUL WURDE AUF FILE   14  UEBERTRAGEN
DER MODUL  ***  MODB  ***  WURDE RICHTIG ANGELAUFEN
DABEI WURDE EIN FELD DER LAENGE      100  UEBERTRAGEN.
DER MODUL  ***  MODA  ***  WURDE RICHTIG ANGELAUFEN
DABEI WURDE EIN FELD DER LAENGE      100  UEBERTRAGEN.
RUECKSPRUNG AUS  MODA  IN MODB WURDE RICHTIG AUSGEFUEHRT.

  AUFRUFFOLGE DER MODULN:  1.)  MODA
                           2.)  MODA
2. MODUL WURDE AUF FILE   15  UEBERTRAGEN
DER MODUL  ***  MODA  ***  WURDE RICHTIG ANGELAUFEN
DABEI WURDE EIN FELD DER LAENGE      100  UEBERTRAGEN.
DER MODUL  ***  MODB  ***  WURDE RICHTIG ANGELAUFEN
DABEI WURDE EIN FELD DER LAENGE      100  UEBERTRAGEN.
DER MODUL  ***  MODA  ***  WURDE RICHTIG ANGELAUFEN
DABEI WURDE EIN FELD DER LAENGE      100  UEBERTRAGEN.
RUECKSPRUNG AUS  MODA  IN MODB WURDE RICHTIG AUSGEFUEHRT.
RUECKSPRUNG AUS MODB IN MODA WURDE ORDNUNGSGEMAESS AUSGEFUEHRT.
```

```
  AUFRUFFOLGE DER MODULN:  1.)  MODB
                           2.)  MODC

2. MODUL WURDE AUF FILE   14  UEBERTRAGEN

DER MODUL  ***  MODB  ***  WURDE RICHTIG ANGELAUFEN

DABEI WURDE EIN FELD DER LAENGE     100  UEBERTRAGEN.

DER MODUL  ***  MODC  ***  WURDE RICHTIG ANGELAUFEN

DABEI WURDE EIN FELD DER LAENGE     100  UEBERTRAGEN.

RUECKSPRUNG AUS  MODC  IN MODB WURDE RICHTIG AUSGEFUEHRT.


  AUFRUFFOLGE DER MODULN:  1.)  MODA
                           2.)  MODC

2. MODUL WURDE AUF FILE   15  UEBERTRAGEN

DER MODUL  ***  MODA  ***  WURDE RICHTIG ANGELAUFEN

DABEI WURDE EIN FELD DER LAENGE     100  UEBERTRAGEN.

DER MODUL  ***  MODB  ***  WURDE RICHTIG ANGELAUFEN

DABEI WURDE EIN FELD DER LAENGE     100  UEBERTRAGEN.

DER MODUL  ***  MODC  ***  WURDE RICHTIG ANGELAUFEN

DABEI WURDE EIN FELD DER LAENGE     100  UEBERTRAGEN.

RUECKSPRUNG AUS  MODC  IN MODB WURDE RICHTIG AUSGEFUEHRT.

RUECKSPRUNG AUS MODB IN MODA WURDE ORDNUNGSGEMAESS AUSGEFUEHRT.

PROGRAMM ORDNUNGSGEMAESS BEENDET,
NACHDEM ALLE AUFRUFE RICHTIG AUSGEFUEHRT WURDEN.
```