Wirtschaftsinformatik II (Theorie)

(Teil 1)

Einführung

Programm

- Eine zur Lösung einer Aufgabe vollständige Anweisung zusammen mit allen erforderlichen Vereinbarungen heißt PROGRAMM.
- Eine Programm setzt sich aus Befehlen zusammen, die von der Zentraleinheit in logischer Reihenfolge abgearbeitet werden.
- Der Vorgang der Erstellung einer derartigen Anweisung heißt PROGRAMMIEREN.

Software

- Die Menge von Programmen, zugehörigen DATEN und notwendigen DOKUMENTATIONEN, die zusammengefaßt erlauben, mit Hilfe eines Computers Aufgaben zu erledigen, nennt man SOFTWARE.

Systemsoftware

- Die Software, die für eine spezielle Hardware oder eine Hardwarefamilie entwickelt wurde, um den Betrieb und die Wartung dieser Hardware zu ermöglichen bzw. zu erleichtern, nennt man SYSTEMSOFTWARE.
- Die Systemsoftware steuert und überwacht die Abwicklung von Anwendungssoftware auf der jeweiligen Maschinenkonfiguration.

Anwendungssoftware

- Die Anwendungssoftware unterstützt die Aufgaben des Anwenders und baut auf der Systemsoftware der verwendeten Hardware auf.
- Individualsoftware vs. Standardsoftware
- Eigenentwicklung vs. Fremdentwicklung
- Offene Systeme vs. Proprietäre Systeme

Programmiersprachen

- Eine Programmiersprache ist eine Künstliche Sprache zum Abfassen von Computerprogrammen.
- Programmiersprachen sollen es ermöglichen, den Lösungsweg (Algorithmus) für eine spezielle Problemstellung in einer der Maschine verständlichen Sprache zu formulieren.

Programmierungsumgebung

- Eine Programmierungsumgebung ist ein Softwaresystem, das eine Sammlung von Werkzeugen, die zur Entwicklung von Software dient, zur Verfügung stellt.
- Wesentliche Komponenten:
 1. TEXTEDITOR: Zur Erfassung des Quelltextes
 2. ÜBERSETZUNGSPROGRAMM: Programm, das Anweisungen in einer höheren Sprache liest, analysiert und in bedeutungsgleiche Maschinenbefehle umwandelt.
 3. DEBUGGER: ermöglicht ein schrittweise Abarbeiten des Programms zu Testzwecken.

4. HILFESYSTEM: erlaubt die kontextsensitive Abfrage von Informationen zur Programmsyntax u. ä.
5. GUI-BUILDER: Modellierung der Benutzeroberfläche

CASE-Systeme

- = Computer Aided Software Engineering-Systeme
- Eine CASE-System ist eine integrierte SOFTWARE-Entwicklungsumgebung, die den gesamten Softwarelebenszyklus unterstützt.

Sprachgenerationen

- Sprachen der ersten Generation: MASCHINENSPRACHE
- Sprachen der zweiten Generation: ASSEMBLERSPRACHEN
- Sprachen der dritten Generation: HÖHERE PROGRAMMIERSPRACHEN
- Sprachen der vierten Generation: 4GL-Systeme
- Alternative Sprachrichtungen:
 - SPRACHEN DER KÜNSTLICHEN INTELLIGENZ (5. Generation)
 - OBJEKTORIENTIERTE PROGRAMMIERSPRACHEN
 - VISUELLE PROGRAMMIERSPRACHEN

Übersetzungsprogramm I

- Ein Programm, das nicht in Maschinsprache abgefaßt ist, wird QUELLPROGRAMM genannt.
- Ein in Maschinsprache vorliegendes Programm (=Zielprogramm des Übersetzungsvorgangs) heißt OBJEKTPROGRAMM (OBJEKTCODE).
- Unter einem ÜBERSETZUNGSPROGRAMM (TRANSLATOR) versteht man ein Programm, das Anweisungen in einer höheren Sprache oder in Assemblersprache liest, analysiert (Optimierung) und in bedeutungsgleiche Maschinenbefehle umwandelt.

Compiler vs. Interpreter

- **Compiler:**
Ein Übersetzungsprogramm, das in einer höheren Programmiersprache abgefaßten Quellcode komplett in Zielanweisungen einer maschinenorientierten Programmiersprache umwandelt (kompiliert).

Das erzeugte Objektprogramm kann für spätere Verarbeitungsläufe abgespeichert werden, so daß vor der Ausführung keine Übersetzung mehr nötig ist.

- **Interpreter:**
Ein Programm zur schrittweisen Übersetzung von Quellcode in lauffähigen Maschinencode, welcher sofort (zeilenweise) ausgeführt wird.

Im Unterschied zum Kompilieren entsteht keine vollständiges, aufbewahrungsfähiges Objektprogramm.

Bei jedem Programmstart erfolgt ein erneuter Übersetzungsvorgang.

Methodik der Programmerstellung

Vorgehensweise

- Die Lösung der Aufgaben, lassen sich in drei Arbeitsschritte aufteilen

- Methode der schrittweisen Verfeinerung (Top-Down-Strategie)
 1. Schritt: PROBLEMAUFBEREITUNG
 - Definition der Aufgabe
 - Festlegen der Programmstruktur nach dem EVA-Prinzip.
 2. Schritt: PROBLEMANALYSE
 - Mit Hilfe der Struktogrammtechnik wird gem. Problemaufbereitung der mögliche Lösungsweg in eine graphische Darstellung umgesetzt.
 - Jeder Programmablauf basiert auf wenigen Grundstrukturen (Sequenz, Auswahl, Wiederholung)
 3. Schritt: PROGRAMMERSTELLUNG
 - In diesem letzten Schritt wird das Struktogramm in in einer höheren Programmiersprache formuliert.
 - Das Quellcode wird in eine Maschinensprache durch den Compiler übersetzt.
 - Fehlerbeseitigung und Programmtest.

Programmerstellung: Konventionen

- kein Unterschied zwischen Groß- und Kleinschreibung
- i. d. R. keine Verwendung deutscher Sonderzeichen (ä,ö,ü,ß) im Quellcode
- jede Anweisung ende i. d. R. mit dem Semikolon (;)
- Punkt als Dezimaltrennzeichen
- ein Kommentar wird im Quellcode durch geschweifte Klammern { } oder durch Zeichenfolgen (*und*) eingerahmt

Beispiel:

```
{Dies ist ein Kommentar}
{* und das ist auch ein Kommentar *}
```

Programmerstellung: Fehlerarten

- Fehler zur Übersetzungszeit (Compiler Fehler)
 1. Beim Übersetzen des Quellcodes in die Maschinensprache werden die syntaktischen und die semantischen Fehler erkannt.
 2. Borland Pascal aktiviert das Editor-Fenster und positioniert den Cursor an die Stelle, an der der Fehler im Quellcode aufgetreten ist.
- Laufzeitfehler (Runtime-Error)
 1. Diese Fehler treten beim Ablauf des Programms auf und führen zum Programmabbruch durch das Betriebssystem.
 2. Beispiel: Division durch 0, Wertebereichsüberschreitungen
- Logische Fehler
 1. Die Aufgabenstellung wird durch das Programm nicht gelöst, weil z. B. ein unbrauchbarer Algorithmus gewählt wurde.
 2. Der Compiler erkennt keine logischen Fehler!

Programmerstellung: Programmierstil

- Lesbarkeit und Verständlichkeit
- Erleichterung der Fehlersuche, Wartung und Pflege
- Stil-Regeln
 1. Optische Trennung der Programmbausteine
 - Leerzeichen
 - Einrücken des Porgrammtextes
 2. Großbuchstaben für Schlüsselwörter, Kleinbuchstaben für Bezeichner, nur der erste Buchstabe groß bei Standard-Funktionen und Standard-Prozeduren

3. „sprechende“ Namen für Bezeichner bzw. Verwendung von anerkannten Namenskonventionen (ungarische Notation von Charles Simonyi)
4. Einsatz von Konstanten
5. Strukturierung der Programme nach Eingabe, Verarbeitung und Ausgabe (EVA-Prinzip)
6. Großzügiger Einsatz von Kommentaren
7. Benutzerfreundlich programmieren

Formaler Aufbau eines Pascal-Programms: Programmkopf (Identifikationsteil)

PROGRAMMKOPF:

- „Überschrift“ für das Programm
- Bestandteile
 - Bezeichner für Programmname
 - Semikolon
- Bezeichner darf im Programm nicht mehr für etwas anderes verwendet werden
- Beispiel: PROGRAM beispiel_1;

DEKLARATIONSTEIL:

- Elemente werden angemeldet, deklariert (konkrete Werte) oder definiert (Beschreibung)
- entsprechender Speicherplatz für Elemente wird reserviert
- programmspezifische Vereinbarungen
 1. Units USES
 2. Konstanten CONST
 3. Datentypen TYPE
 4. Variablen VAR
 5. Prozeduren PROCEDURE
 6. Funktionen FUNCTION

VERARBEITUNGSTEIL:

- Sammlung von Anweisungen
- beginnt mit dem reservierten Wort BEGIN
- mit dem reservierten Wort END und dem folgenden Punkt wird das Ende des Verarbeitungsteils (und damit des gesamten Programms) signalisiert
- die einzelnen Anweisungen werden durch Semikolon getrennt (Semikolon ist ein Anweisungstrennzeichen und kein Anweisungsendezeichen)

SYNTAXDIAGRAMME

- Durch die Syntax einer Programmiersprache wird der sprachlich grammtikalisch richtige Aufbau eines Programms beschrieben.
- gerichteter Graph, bei dessen Durchlauf automatisch syntaktisch richtige Programme entstehen
- Notation

Für die symbolhafte Darstellung von Syntaxdiagrammen gelten folgende Vereinbarungen:

1. Ein **Sprachelement** (spezielle Symbole, Bezeichner, Zahlen, Zeichenketten und Labels), dessen Zeichenfolge und Bedeutung **in der Programmiersprache** eindeutig **festgelegt** ist und nicht mehr weiter erklärbar ist, wird in einem abgerundeten Kästchen abgebildet.

2. In einem eckigen Kasten stehen Sprachelemente, die eine komplexe, **syntaktische Einheit** darstellen und durch weitere Syntaxdiagramme erklärt werden müssen.
Die Reihenfolge der Sprachelemente wird durch Pfeile gekennzeichnet.
3. Die Symbole (Elemente) werden durch Linien (Pfeile), die die Ablaufrichtung anzeigen, miteinander verbunden.

Strukturierte Programmierung

Ziele der strukturierten Programmierung

- Unterstützung einer methodischen Vorgehensweise beim Programmentwurf
- lineare Form des Programmcodes ohne unbedingte Sprünge (kein „Spagetticode“)
- Ergebnis sind hierarchisch gegliederte Programmstrukturen
- Steigerung der Lesbarkeit / Übersichtlichkeit
- Steigerung der Zuverlässigkeit
- Verbesserung der Wartung und Pflege

Prinzipien der strukturierten Programmierung

- Prinzipien der schrittweisen Verfeinerung:
Strukturierung des Programms durch verschiedene Abstraktionsebenen
- Prinzipien der Modularisierung:
Gesamtaufgabe wird solange weiter in logisch zusammenhängende Teilaufgaben zerlegt, bis kleine überschaubare Einheiten (Module) entstehen (Top-Down-Methode)
- Prinzip der linearen Kontrollstrukturen:
Jeder Programmablauf ist auf die Grundstrukturen
 1. Sequenz
 2. Auswahl und
 3. Wiederholung
 zurückzuführen.

(Siehe Struktogrammtechnik S. 31)

Struktogrammtechnik: Sequenz

- mehrere Anweisungen sind hintereinander auszuführen
- Abarbeitung erfolgt von oben nach unten
- Beispiel: Nachdem der Kundenauftrag eingegangen ist, wird die Ware zusammengestellt und danach verschickt.
Anschließend wird die Rechnung angefertigt und versandt.

Struktogrammtechnik: Auswahl I

- Anweisungen werden in Abhängigkeit von bestimmten Bedingungen ausgeführt
- Ergebnis eines booleschen Ausdrucks steuert den weiteren Programmablauf.
- 3 verschiedene Auswahl-Konzepte
 1. einseitige Auswahl (bedingte Verarbeitung)
 2. zweiseitige Auswahl (einfache Alternative)
 3. Mehrfachauswahl (mehrfache Alternative)

Struktogrammtechnik: Auswahl II

- **Einseitige Auswahl.**
Ist die Bedingung erfüllt (d. h. wahr), wird der linke Strukturblock (Anweisung 1) verarbeitet, ansonsten wird der Programmablauf mit dem nachfolgenden Strukturblock fortgesetzt.
- Beispiel: Wenn ein Kunde im vergangenen Jahr Waren bezogen hat, erhält er eine Weihnachtskarte, hat er keine Waren bezogen, so passiert nichts.
- **Zweiseitige Auswahl:**
In Abhängigkeit von dem Auswertungsergebnis (wahr oder falsch) der Bedingung kommt entweder der linke Strukturblock (Anweisung 1) oder der rechte Strukturblock (Anweisung 2) zur Verarbeitung.
- Beispiel. Wenn ein Kunde im vergangenen Jahr Waren bezogen hat, erhält er eine Weihnachtskarte, sonst wird er im Kundenverzeichnis gelöscht.

Struktogrammtechnik: Mehrfachauswahl

- Abhängig von einer Steuerungsgröße (Selektor) tritt eine Option aus einer Auswahl von Alternativen ein. Hierbei gelangt die Option zur Ausführung, deren Marke mit dem aktuellen Wert des Selektors übereinstimmt. (CASE-Anweisung)
- Beispiel: Wenn ein Kunde im vergangenen Jahr Waren größer als 1000 DM bezogen hat, erhält er einen Treuebonus von 10 %, hat er Waren zwischen 500 Dm und 1000 DM bezogen, erhält er einen Bonus von 5 %, hat er Waren bis 500 DM bestellt, erhält er einen Bonus von 2 %, sonst wird er im Kundenverzeichnis gelöscht.

Strukturgrammtechnik: Wiederholung I

- Eine oder mehrere Anweisungen sollen in Abhängigkeit von einer Bedingung wiederholt werden oder für eine gegebene Anzahl von Wiederholungen durchlaufen werden.
- Ergebnis eines booleschen Ausdrucks steuert den weiteren Programmablauf
- 3 verschiedene Wiederholungskonstrukte
 1. Wiederholung mit Abfrage vor jedem Wiederholungsdurchlauf
 2. Wiederholung mit Abfrage nach jedem Wiederholungsdurchlauf
 3. Wiederholung mit fester Wiederholungszahl

Strukturgrammtechnik: Wiederholung II

- Wiederholung mit Abfrage vor jedem Wiederholungsdurchlauf:
Es erfolgt eine Prüfung **VOR** der abzuarbeitenden Anweisung, ob die Verarbeitung zu wiederholen bzw. überhaupt ein erstes Mal durchzuführen ist.
Ist die Bedingung nicht mehr erfüllt wird hinter der zu wiederholenden Anweisungsfolge fortgefahren.
- Beispiel: Solange ein Kunde in jedem Jahr mindestens eine Bestellung aufgibt, wird ihm der Jahreskatalog kostenlos zugesandt.
- Wiederholung mit Abfrage nach jedem Wiederholungsdurchlauf:
Es wird **NACH** jedem Durchlauf der Schleife geprüft, ob die Verarbeitung wiederholt werden soll.
Ist die Bedingung erfüllt, wird mit der nächsten Anweisung fortgefahren.
- Beispiel: Wiederhole Aktionen zur Neukundenwerbung bei dem potentiellen Kunden, bis der Kunde die erste Bestellung erteilt.

Struktogrammtechnik:

Wiederholung mit fester Wiederholungszahl

- Bei einer **ZÄHLSCHLEIFE** ist die Anzahl der Wiederholung von Anfang an bekannt.
- Ein Schleifenzähler läuft von einem Anfangswert- zu einem Endwert.
- Vor jedem Durchlauf erfolgt die Abfrage, ob der Endwert des Schleifenzählers erreicht ist.
- Beispiel: Bei ausstehenden Zahlungen werden 3 mal Mahnungen verschickt, bis die Unterlagen an die Rechtsabteilung übergeben werden.

Einfache Datentypen (Allgemeines)

- *Variablen* sind Datenfelder, die während eines Programmablaufs *unterschiedliche Werte* annehmen können. D. h. sie lassen sich verändern.
- Jede Variable hat einen Namen, Inhalt (Wert) und einen Datentyp.
- Eine mögliche *Einteilung* der Datentypen ist die in *unstrukturierte und strukturierte* Datentypen.
- *Datentyp* legt die *Menge von Werten (Wertebereich)* fest, die eine Variable annehmen kann.
- Zu jedem Datentyp gibt es *Operationen*, mit denen Daten des Datentyps *verknüpft* bzw. *manipuliert* werden können.
- Die Wertzuweisung, ist bei jedem Datentyp zulässig; arithmetische Operationen können z. B. nur auf Zahlen angewendet werden.
- Weist der Wertebereich einer Variablen eine Ordnung seiner Elemente auf, d. h. es kann ein Vorgänger und ein Nachfolger eines jeden Elements bestimmt werden, so ist dieser Datentyp *ordinal*.
- Zu den ordinalen Datentypen zählen u. a. char, boolean und interger.

Konstanten

- Datenfelder, die während eines Programmablaufs einen konstanten Wert haben (darf im Programm nicht verändert werden)
- der Compiler erkennt jeweils selbständig, von welchem Datentyp die jeweiligen Konstanten sind.

Variablen

- Datenfelder, die während eines Programmablaufs veränderbare Werte annehmen können.

Datentypen – Systematisierung (Zusammensetzung)

- Ein Datentyp:
 1. legt eine Menge von Werten (Wertebereich) sowie
 2. eine Sammlung von darauf zugeschnittenen Operationen fest.

(Klassifikationskriterium: Zusammensetzung siehe S. 37)

Datentypen – Systematisierung (Speicherverwaltung)

- Dynamische Variablen erhalten ihren Speicherplatz nicht zur Übersetzungszeit, sondern erst zur Laufzeit.
- Das Ziel dynamischer Variablen ist, keinen Speicherplatz bei einer unbekanntem Zahl von zu speichernden Datenobjekten zu verschwenden.

(Klassifikationskriterium: Speicherverwaltung siehe S. 37)

Datentyp Integer

- ist eine Untermenge der ganzen Zahlen (positive und negative)
- Wertebereich: -32.768 (-215) ..32.767 (215-1)
In der Programmierung ist die Menge der ganzen Zahlen aufgrund der begrenzten Darstellungsmöglichkeiten des Rechners nicht unendlich.
- Speicherplatz: 16 Bit
- Operationen:
Rechenoperationen Addieren, Subtrahieren und Multiplizieren können bei dem Datentyp problemlos angewendet werden, da das Ergebnis solcher Operationen wiederum ganzzahlig ist.
In Turbo Pascal gibt es für Größen des Datentyps integer zwei spezielle Arten der Division.
DIV-Operator: Als Ergebnis ergibt sich der ganzzahlige Anteil des Divisionsergebnisses
MOD-Operator: Wird verwendet, um den Rest einer integer-Division zu erhalten.
Bsp: 7 MOD 2 liefert den Wert 1 (Rest der ganzen Division)
- Integerüberlauf oder Laufzeitfehler bei Wertebereichsüberschreitung.
- Beispiel: VAR anz_stud, alter: integer;

Datentyp Real

- ist eine Untermenge der positiven und negativen reellen Zahlen.
- Wertebereich: +-2.9E-39..+-1.7E+38
In der Programmierung ist eine lassen sich nicht beliebig große positive bzw. negative Zahlen darstellen und auch nicht beliebig kleine Zahlen, die in der Umgebung des Wertes Null liegen.

Der Datentyp real hat als Wertebereich alle darstellbaren reellen Zahlen, die im Positiven Intervallbereich von 2.9E-39 bis 1.7E+38 liegen. Der Wert 0.0 ist auch definiert.
- Wir unterscheiden bei real-Werten die Bildschirmdarstellung und Speicherung in Exponentialschreibweise bzw. Gleitkommenschreibweise.
Turbo Pascal Compiler arbeitet mit der Gleitpunktdarstellung. Bei dieser Schreibweise werden die Zahlen in Form von Realwert = Mantisse * Basis (10) hoch Exponent dargestellt.

Datentyp Char I

- Datentyp character ermöglicht die Eingabe eines beliebigen Zeichens aus dem ASCII-code.
- Das sind Buchstaben, Ziffern, Steuerzeichen, Sonderzeichen
- Wertebereich: 256 Zeichen des ASCII-Zeichensatzes
- Speicherplatz: 8 Bit
- Zuweisung erfolgt in Hochkommata
- Beispiel: VAR eingabe, geschlecht. char;
- Ausschnitt aus dem ASCII-Zeichensatz

- Identisch mit dem entsprechenden ANSI-Windows-Zeichensatz

Datentyp Boolean I

- Die Wahrheitswerte „true“ und „false“ können einer Variablen vom Datentyp boolean zugeordnet werden.
- Diese booleschen Variablen bzw. booleschen Aussagen lassen sich durch Operationen verknüpfen.
- Die Grundoperationen sind die logischen Verknüpfungen „and“, „or“ und „not“.
- Die Ergebnisse logischer Verknüpfungen, deren Operanden vom Typ boolean sein müssen, sind wieder vom Typ boolean.
- Wertebereich: false, true
- Speicherplatz: 1 Byte
- Operationen: Siehe Vorlesungsbeilage S. 40
- Datentyp Boolean ist ein ordinaler Datentyp
Es gilt: - fals < true
 Ord (false) = 0
 Ord (true) = 1
Beispiel: VAR ist_ok, rechnung_bezahlt: boolean;
- Rangfolge bei der Abarbeitung logischer Verknüpfungen
 1. ()
 2. NOT
 3. AND
 4. OR / XOR
 (Beispiele Siehe S. 40)

Aufzählungstyp

In Pascal können Datentypen vom Programmierer selbst definiert werden, wie z. B. der Aufzählungstyp.

Die Deklaration erfolgt vor der Variablendeklaration im Deklarationsteil des Programms.

Nach dem reservierten Wort TYPE wird für den neu zu definierenden Datentyp ein Name (Bezeichner) vergeben.

Dann wird sein Wertebereich festgesetzt.

Datentypvereinbarung sollte nach der Konstantendeklaration und vor der Variablendeklaration (falls vorhanden) erfolgen.

CONST	Definition
TYPE	Definition
VAR	Definition

Ein *selbstdefinierter Datentyp* wird als *Aufzählungstyp* bezeichnet, wenn in seinem Wertebereich die für eine Variable vorgesehenen Werte der Reihe nach aufgeführt werden.

- Verwendung findet bei Datenobjekten, die eine Menge von Einzelobjekten darstellen
- Ziel: Verwendung von Problemspezifischen, übersichtlichen Datentypen, die keinen Speicherplatz verschwenden.
- Wertebereich: wird durch Aufzählung selbst definiert
- Speicherplatz: 1 Byte (<256 Elemente)
- Operationen: Standardfunktionen für ordinale Datentypen:

Ord (X), Succ (X), Pred (X)

- Beispiel: VAR wochentage: (Mo, Di, Mi, Don, Fr, Sa, So);
farbe: (rot, gruen, blau, weiss, gelb, schwarz);

Teilbereichstyp

Dieser Datentyp beschreibt einen Teilbereich eines ordinalen Wertebereichs (Grundmenge), der vom Programmierer durch die Angabe seiner begrenzenden Elemente festgelegt wird.

- aus jedem ordinalen Datentyp (Basistyp) kann man eine Teilmenge des Wertebereichs als eigenen Datentyp definieren.
- Ziel: Verwendung von problemspezifischen, übersichtlichen Datentypen, die keinen Speicherplatz verschwenden (z. B. bei Eingabebeschränkungen)
- WERTEBEREICH: wird durch kleinsten und größten Wert explizit definiert
- SPEICHERPLATZ: 1 Byte (<256 Elemente)
- OPERATIONEN: Operationen des Basistyps
- Beispiel: VAR noten: 1...6;
eingabe: 'A'... 'D';

Type-Anweisung

- zur Definition problemspezifischer, selbstdefinierter Datentypen, die dann zur Variablendeklaration benutzt werden können
- Datentyp wird mit einem Namen bezeichnet und ein Wertebereich wird festgelegt, aus dem eine Variable dieses neudefinierten Datentyps einen Wert annehmen kann
- nach dem Konstantendeklarationsteil (CONST) und vor dem Variablendefinitionsteil (VAR).
- Beispiel: TYPE dm_typ: real;
VAR netto_preis, brutto_preis: dm_typ;

Aufzählungstypen / Teilbereichstypen unter Verwendung der TYPE-Anweisung

(Siehe Vorlesungsbeilage S. 43)

Standardfunktionen

(Siehe Vorlesungsbeilage S. 44)

USES-Anweisung

- spezielle Bibliotheken (UNITS), die eine Sammlung von vordefinierten Konstanten, Datentypen, Variablen, Prozeduren und Funktionen enthalten, die nicht zum Standard-Befehlsumfang von PASCAL gehören.
- Beispiel: USES wincrt;