„Grundlagen Wissenschaftlichen Arbeitens“

Proseminar Programmiersprache
Wintersemester 2005
(Prof. Jens KNOOP)

Vor-Nach Name: Emre ÖZTÜRK

Matr.Nr : 0327616

Kennzahl : 534

emoz159@yahoo.com

Inhalt:

Emre ÖZTÜRK 0327616 534 emoz159@yahoo.com

Was ist Turing?

Turing ist eine universelle Programmiersprache. Dieses Programm ist spezifisch für das Unterrichten der Konzepte der Informatik bestimmt.

Diese Sprache ist entwickelt worden, um Unabhängigkeit des bestimmten Computers zu sein. Die Unabhängigkeit bedeutet hier, die Orientierung der Problemen, die die Personen wünsch tun. Es bezeichnet eine einfache Syntax zu erlernen, die die Personen die starke Fehlerprüfung mitteilen. Damit kann man die Programmierung einfacher bilden.
Weil es in dieser Sprache keinen direkten Zugriff zu der Hardware geben kann, ist Turing eine vollständige „sichere“ Programmiersprache.

Das Klima Turing

Turing wurde entworfen, um die Programme ohne die Ausgabenzeit leicht zu schreiben. Das Klima Turing ist auch für einfache Leitung in den Schulenetzen bestimmt.

Eigenschaften Des Klimas:

- Erlaubt Lehrern, zwischen einem einfachen Einzelnfenstermodus oder Multifenstermodus für erfahrener Kursteilnehmer zu wählen
- Eintastenprogrammmeinrückung
- Eine Tastenkompilation und Durchführung
- Syntaxfarbton
- Einfach Fehlermeldungen zu verstehen
- Direkter Zugriff zum PC-Ähnlichkeitstor

Die Geschichte der Entwicklung der Turing Programmierung

Die Programmiersprache ist im Jahr 1982 von Richard C Holt und James Cordy an der Universität von Toronto (Kanada) entwickelt. Turing ist ein Nachkomme von Euclid. Die Eigenschaften der Turing der sauberen Syntax und Unabhängigsemantik ist genau wie Euclid.

Überblick „The Turing Programming Language“

Weil Turing fast alle Eigenschaften des Pascal enthält, kann für ein Superpascal beschrieben werden.

Diese Programmiersprache besteht aus Aussage und Erklärungen und diese erhalten Ausdrücke, die Anweisungen zu den Variablen, Schleifen, wenn Ausdrücke und Fallausdrücke anschließen. Erklärungen werden verändert, um Einzelteile wie Variablen als subprogramms zu nennen. Ausdruck in Turing sind ganz wie die im Pascal mit den folgenden bemerkenswerfen Ausnahmen. Erstes Turing muss mehr Operatoren (Exponentiation) und =(Implikation) sein. An zweiter Stelle erlauben Turing’s Vorausgehenrichtlinien bestimmte Ausdrücke, die von Pascal verboten werden. zB: in Turing kann man „i=1, orj=1“ schreiben, während die gleichwertigen Ausdrücke Exraklammern im Pascal erfordern würden.

Das Programm

- ### Wille als Vereinbarung nicht als Regel von Turing Sprache

  Wir starten alle Programme mit einer Linie wien diesem % das Programm „opus1“, welches das Namenopus eins ist, wir bildeten, um das frist komplette Programm zu beschreiben.(der Wort Opus ist für „Arbeit“ lateinisch) Sie müssen einen Bezeichner bilden, den Sie sich mögen. Er sollte nicht freie Räume oder Sonderzeichen enthalten.

  Er dient, anzuzeigen, daß es der Anfang eines Programms ist und gibt den Namen, dem wir pflegen, das Programm in unserer Unix Akte einzuordnen. Kommt jetzt der grosse Moment für ein komplettes Programm % das "gesetzte" Programm "opusl1 2+3 =", 2+3 dort, das es ist, unser Opus Nr. eine, ein komplettes Turing Programm. Der Körper des Programms von ist ein einzelnes statment, die Ablegeanweisung.

- Design-Ziele für die Turing Programmiersprache

Das Design von Programmiersprachen bleibt eine kunst ein bestes, und jeder möglicher Versuch, sie auf einer einfachen, quantitativen Disziplin zu verringern wird auszufallen gesprungen. Gleichzeitig, gibt es viele Richtlinien und Grundregeln, die dem Entwerfer helfen können, eine bessere Sprache zu produzieren. Die Designziele für die Turing Sprache werden, beschrieben zusammen mit, wie diese Ziele erreicht wurden.

Wir verzeichnen Ziele für Turing mit ihren Hauptverzweigungen in den Abschnitten, die sich daran konzentrieren:

- Generallity
- Mühelosigkeit des Lernens
- Zuverlässigkeit
- mathematische Präzision
- Leistungsfähigkeit
- Globalen Ziele

Generallity

Turing ist entworfen, um eine universelle Programmiersprache zu sein. Dieses Designziel bedeutet, daß Turing die Sprache der Wahl für eine breite Kategorie Anwendungen möglicherweise sein sollte. Im Auszug ist das Konzept des Allgemeinen schwierig oder unmöglich zu definieren, und es ist nicht angemessen zu versuchen, eine Sprache zu entwerfen, die zu allen Zwecken gut ist. Die Annäherung, die im Design von Turing genommen wurde, sollte das Dienstprogramm anderer universeller Sprachen, so, als Pascal und PL/I betrachten, und diesen Einblick verwenden, um die Breite der Anwendung von Turing beim Beibehalten anderer Sprachenziele wie Mühelosigkeit des Lernens und des implementability zu maximieren.

Mühelosigkeit des Lernens

Turing ist entworfen, um die Mühelosigkeit zu sein, zum zu erlernen. Dieses Designziel verlangt Einfachheit und Bequemlichkeit des Gebrauches. Es erfordert Spracheneigenschaften, die kurzen, freien Erklärungen zu haben.

Niveaus der Beherrschung

Niveaus der Beherrschung bedeutet, daß ein Anfänger in der LageSEIN muß, mit einer Sprachenteilmenge zu beginnen und in zunehmendem Maße größere Teilmengen dann zu erlernen. Es sollte nicht notwendig sein z.B. Details der Formatierung Linien des Ausganges zu erlernen, bevor man ein Programm laufen läßt.

Ein Ziel für Turing war, als BASIC zu erlernen einfacher zu sein. Rulle:Easier als BASIC. Das Ziel ist das für Konzepte und Algorithmen, die BASIC gut anfaßt, die entsprechende Teilmenge von Turing sollten einfacher sein zu erlernen und zu verwenden. Turing ist, um das einfaches Lernen und Gebrauch auf dem Niveau von BASIC zu erlauben, entworfen und wechselwirkende Interpreten zu erlauben. Gleichzeitig ist Turing eine in hohem Grade strukturierte Sprache, die zur Verfügung stellt hoch entwickelte Eigenschaften für das Lösen einer reichen Kategorie programmierenprobleme.

Zuverlässigkeit

Turing ist entworfen, um reliabilty zu fördern. Dieses Designziel bedeutet, daß Turing dem Programmierer in sich entwickelnder Software helfen sollte, die zuverlässig seine spezifizierten Zwecke erreicht.

Das Euclid und gleichzeitigen die Euclid Sprachen mit Pascal als Grundlage und Turing mit Euclid als Grundlage, haben fortgesetzt die Entwicklung in Richtung zur Hilfsprogrammzuverlässigkeit. Turing ist entworfen, um Zuverlässigkeit in drei Hauptmöglichkeiten zu fördern. Die ersten zu den Weisen werden in diesem Abschnitt besprochen. Zuerst ist Turing entworfen, damit möglicherweise krank-verwendetes Konstruieren minimale Störungen der minimalen Schwierigkeit verursacht. Zweitens ist Turing entworfen, um Steuerprogramm Kompliziertheit zu helfen und Programm erklärlichkeit folglich zu maximieren; understandability erhöht Zuverlässigkeit, weil erhöht Primärquelle des Software-Ausfalls ist unsere Unfähigkeit, große Programme während der Programmwartung zu begreifen. Drittens wie in einem neueren Abschnitt besprochen wird, Turing`s stellt formale Definition eine consructive Annäherung zum Produzieren der korrekten Programme zur Verfügung.

Mathematische Präzision

Turing ist entworfen, um eine mathematisch exakte (formale) Definition zu haben. Eine Antriebskraft hinter desing Turing`s war: Richtlinie. Alle Resultate folgen von der Sprachenspezifikation. Dieses reguires, die alles, das über ein Durchführung Turing Programm beobachtet werden kann (ausgenommen Fragen der Leistungsfähigkeit) von der Sprachendefinition vorhersagbar sein sollte. (diese Richtlinie führt zu das Konzept der zuverlässigen Durchführung.) Ein Grund Turing wurde war einfach das intellektuelle challange des Tuns es formalisiert. Wir wollten unsere Fähigkeit demonstrieren, das mathematization einer bequemen, leistungsfähigen, universellen Programmiersprache zu produzieren.

Leistungsfähigkeit

Turing ist für Leistungsfähigkeit bestimmt. Dieses Designziel bedeutet daß Turing Programme, klein und zu fasten wenn, so klein und fasten, wie in den maschinennahen Programmiersprachen wie C möglich seien Sie. Die grundlegende Tatsache, die die zu Leistungsfähigkeit führt, ist, daß Turing eine Verfahrens (zwingende) Sprache ist, in der Sprachenfamilie einschließlich Fortran und Pascal. Als Regel Sprachen in dieser Familie Brache: Richtlinie. Direkt durchführbar. Dies heißt, daß auf standart Computer architechtures, die meisten Spracheneigenschaften nicht bedeutende Laufzeitunterstützung erfordern (bemerkenswerte Ausnahmen schließen input/outputfacilities, mathematische Unterprogramme und Zeichenkettehandhabung ein).

Globale Ziele

Dort sind globales Zielsprachendesign, das nicht leicht kategorisiert werden kann einschließlich: Coherence.The Sprache sollte in ein peasing Wesen zusammen passen. Enjoyment.People sollte Spaß haben, die Sprache Acceplability.People zu verwenden sollte bereit sein zu versuchen annehmen die Sprache. Das Design von Turing versuchte, diese Ziele durch konstante Berichtspracheneigenschaften zu treffen ihr Dienstprogramm ihre Einfachheit, ihre Annehmbarkeit und ihre intredactions.

Eigenschaften von Turing

Turing umfasst alle Standardausrüstungen einer Verfahrensprogrammiersprache. Es umfasst:

- Bequeme Zeichenketten und Input/Output,
- wenn Aussagen mit elsifklauseln,
- Fallaussagen mit anders Klauseln,
- Schleifenaussagen mit Ausgängen,
- Module mit Import und Export,
- dynamische Reihen und Parameter,
- Behauptungen (erklären Sie, vor, Pfosten und unveränderlich),
- schreiben Sie Übergangsfunktionen (wie Zeichenkette zu realem),
- Erzeugung der gelegentlichen Zahl,
- Exponentiation und
- Laufzeitkonstanten.

Programmbeispiel mit einer Schleife

Ist hier ein komplettes Programm Turing, das wiederholt eine gelegentliche Zahl von 1 bis 6 auswählt, bis es 6 auswählt.

```
% Roll a die until you get 6.  (This line is a comment)
var die : int
loop
    randint (die, 1, 6)
    exit when die = 6
    put "This roll is ", die
end loop
put "Stopping with roll of 6"
```

Graphik Eigenschaften Turings

Turing bildet es sehr einfach, damit Kursteilnehmer Eigenschaften benutzen und umfassen:

- Zeichnende Primitive (Linien, Vierecke, Ovale, Sterne und Ahornholz verläßt),
- Ladengraphikakten (BMP und Macintosh PICT),
- einfacher Animation,
- Ton und
- Musik.

Programmbeispiel Mit Graphiken

Ist hier ein komplettes Programm Turing, das nach dem zufall farbige Sterne in einem Nachthimmel zeichnet.

```
    var x, y, clr : int                          % The location and color of the star
    drawfillbox (0, 0, maxx, maxy, black)          % Make the window black
    drawline (0, 100, maxx, 100, white)            % Draw the horizon
    for i : 1 .. 40
        randint (x, 0, maxx)                       % Set the x position
        randint (y, 100, maxy)                     % Set the y position
        randint (clr, 0, maxcolor)                 % Set the color
        drawfillstar (x, y, x + 20, y + 20, clr)   % Draw the star
    end for
```

Dieses Beispiel zeigt auch das *maxx* , *maxy* , *maxcolor*- Funktionen, die es einfach, Programme für jede mögliche Schirmauflösung und irgendeine Plattform zu schreiben bilden. Kursteilnehmer können Programme schreiben, die auf ihren Macintosh zu Hause so leicht wie ihr PC an der Schule laufen.

Beispiele der Eigenschaften von Turing

2 Kurze, Komplette Programm Beispiel

Eine Reihenfolge von Hello und Goodbye seperate ausgeben Linien, das komplette Turing Programm

```
put" Hello"
is
loop
    put "Hello"
    put "Goodbye"
 endloop
```

es ist, zum das aquivalent im Pascal zu betrachten ein lehrreiches

```
  program test (output);
        begin
            while to do
                  begin
                       writeln("Hello");
           writeln("Goodbye")
        end
    end .
```

Dynamische Reihen

Ist hier ein Beispiel, in dem die Größe einer Reihe notknown an Kompilierzeit ist: var n: interne gesetzte "Gebengröße" erhalten n var a: Reihe 1. . n der dynamischen Reihe des realen... Gebrauches... Variable a ist eine dynamische Reihe, deren Größe determinedat Durchlaufzeit durch den Wert von n ist, das in der dritten Linie eingegeben wird.

Summieren einer Reihe

Das folgende Beispiel veranschaulicht Funktionen, dynamische arrayparameters und für Aussagen: gelesene % und summieren eine Liste von Zahlen, die var arraysize: intern erhalten Sie;arraySize %, die innen gelesen werden, sortieren für Reihe einen var a: Reihe 1 arraysize von realen % Laufzeitlimitfunktion sum@: Reihe 1. * von realem): reale var Gesamtmenge: real: = 0.0 für J: 1 upper(b) % für jede Feldelementgesamtmenge: = Gesamt- + b(i)ende für Resultat Gesamtende Summe. rechnenelemente der Reihe ein gesetzte "Summe der Reihe:", sum(a)

Formale Definition von Turing

Eins der Hauptdesignziele von Turing war, daß thelanguage formaler Definition zugänglich sein sollte. Aseach Spracheneigenschaft war, es war scrutinizedto überprüfen entworfen, ob vorhandenes Formalisierung einer Sprache techniquescould an ihm es ermöglichend producea zur formalen Definition angewendet wird, die im Wesentlichen alle Aspekte der Sprache spezifiziert. Wir jetzt besprechen kurz thepurpose dieser Definition und der Form, die es nimmt. (für eine umfangreichere Diskussion über diese Themen und mehr Hinweise, kann der Leser möchte sehen

Formale Syntax von Turing

Die formale Definition von Turing wird in die zwei Teile geteilt: Syntax und Semantik. Die Syntax von Turing spezifiziert effektiv die "Form" von Programmen, ohne über ihre Bedeutung betroffen zu werden. Turings formale Syntax wird durch eine Boolesche Funktion definiert, die syn genannt wird, das eine Zeichenkette s der Buchstaben abbildet, um auszurichten, wenn die Zeichenkette ein Turing Programm ist. syn(s) =, +flex(s) u. cfs(s) u. cc(s) Dieses wird in Tabelle 2 veranschaulicht. Einfach sprechend, prüft lex, um zu sehen, wenn Wörter im Programm wohlgeformt sind, CFS- prüft, um zu sehen, wenn diese Wörter zusammen in eine angemessene Reihenfolge aufgereiht werden, und cm überprüft, um zu sehen, wenn die Reihenfolge eine angemessene Bedeutung gegeben werden kann. Die cm Überprüfung wird groß mit der Art betroffen z.B., die überprüft verhindert einen Zeichenkettewert an einer Ganzzahlvariable zugewiesen werden. Eine Zeichenkette, die lex erfüllt, CFS- und cm wird betrachtet, a (zugelassenes) Turing Programm zu sein. Wir beschreiben jetzt die Methoden, die verwendet werden, um lex, CFS- zu definieren, und cm Turings für mal Syntax die formale Definition von Turing wird in die zwei Teile geteilt: Syntax und Semantik. Die Syntax von Turing spezifiziert effektiv die "Form" von Programmen, ohne über ihre Bedeutung betroffen zu werden. Turings formale Syntax wird durch eine Boolesche Funktion definiert, die syn genannt wird, das eine Zeichenkette s der Buchstaben abbildet, um auszurichten, wenn die Zeichenkette ein Turing Programm ist. syn(s) =, +flex(s) u. cfs(s) u. cc(s) Dieses wird in Tabelle 2 veranschaulicht. Einfach sprechend, prüft lex, um zu sehen, wenn Wörter im Programm wohlgeformt sind, CFS- prüft, um zu sehen, wenn diese Wörter zusammen in eine angemessene Reihenfolge aufgereiht werden, und cm überprüft, um zu sehen, wenn die Reihenfolge eine angemessene Bedeutung gegeben werden kann. Die cm Überprüfung wird groß mit der Art betroffen z.B., die überprüft verhindert einen Zeichenkettewert an einer Ganzzahlvariable zugewiesen werden. Eine Zeichenkette, die lex erfüllt, CFS- und cm wird betrachtet, a (zugelassenes) Turing Programm zu sein. Wir beschreiben jetzt die Methoden, die verwendet werden, um lex, CFS- und cm zu definieren.

Plattformen Turing und Systemanforderungen

Versionen von Turing bestehen für die Microsoft- Windows, Apple- Macintosh und Linuxplattformen. Die Sprache Turing ist zwischen Plattformen identisch und Akten sind gerader ASCII und ermöglichen Kursteilnehmern, Programme von Haus zu Schule unabhängig davon Plattform seamlessly zu bringen. Systemanforderungen Turings sind ein Tief, da möglich, damit Schulen jedes vorhandene Labor verwenden können. Turing wird entworfen, um entweder über ein Netz zu laufen oder angebracht auf einzelne Maschinen.

Microsoft Windows	Windows 95/98/Me/NT/2000/XP RAM 16MB Harter AntriebscRaum Mb 6
Apple Macintosh	System 7,1 RAM 16MB Harter AntriebscRaum Mb 4
Linux	RAM 16MB Harter AntriebscRaum Mb 6

Zusammungfassung

Diese Sprache ist entwickelt worden, um Unabhängigkeit des bestimmten Computers zu sein. Die Unabhängigkeit bedeutet hier, die Orientierung der Problemen, die die Personen wünsch tun. Wir sagen, daß Turing ein anteilig einfaches zu erlernen ist. Da ein Problem Sprache orientierte, wird es mit Problemen numerischen Berechnungen wie auftritt in den wissenschaftlichen Anwendungen einer Technik sowie mit der alphabetischen Manipulation der Informationen betroffen, die durch Geschäft und humanistische Anwendungen erfordert wird. Außerdem kann es für Programmiersystem-Software verwendet werden. es ist eine Verfahren orientierte Sprache, die bedeutet, daß sie zu den expres Algorithmen entworfen war.

REFERENZEN:

R.C.Holt
J.N.P.Hume

- "Introduction to Computer Science usign the Turing Programming Language"

  Richard C.Holt
  Philip A. Matthews
  J.Alan Rosselet
  James R.Cordy

- "The Turing Programming Language Design and Definition"

- Im Web : http://portal.acm.org/