2.1 NAMES

Introduction

Within programming a variety of items are given descriptive names to make the code more meaningful to us as humans. These names are called “Identifier Names”. Constants, variables, type definitions, functions, etc. when declared or defined are identified by a name. These names follow a set of rules that are imposed by:

- The language’s technical limitations
- good programming practices
- common industry standards for the language

Imperative programming languages are abstraction of the underlying von Neumann computer architecture.

- Memory : stores both instructions and data
- Processor: Provides operations for modifying the contents of the memory

Abstraction for memory is called as variables. Sometimes abstraction is very close to characteristics of cells.

e.g. Integer – represented directly in one or more bytes of a memory

In other cases, abstraction is far from the organization of memory.

e.g. Three dimensional array. Requires software mapping function to support the abstraction

A variable is characterized by a collection of properties (attributes)

- Name
- Address
- Value
- Type
- Scope
- Lifetime

Variables, subprograms, labels, user defined types, formal parameters have names.

Design Issues:

- Maximum length of a name
- Case sensitive or case insensitive
- Special words (reserved words, keywords or predefined names)

Name forms:

- Name is a string of characters
- Every language has a different string size.
- Earliest languages : single character
- Fortran 77 : up to 6 characters
- Fortran 95 : 31 characters
- C89 : no limit but only first 31 are significant
- Java, C#, Ada : no limit
- C++ : no limit , but sometimes implementors have

• Names in most PL have the same form:

A letter followed by a string consisting of letters, digits, and underscore characters

• Today "camel" notation is more popular for C-based languages (e.g. myStack)

• In early versions of Fortran – embedded spaces were ignored. e.g. following two names are equivalent

Sum Of Salaries

SumOfSalaries

Case sensitivity:

In many languages (e.g. C-based languages) uppercase and lowercase letters in names are disctinct

e.g. rose, ROSE, Rose

• Problem for readability – names look very similar denote different entities

• Also bad for writability since programmer has to remember the correct cases

e.g. Java method parseInt for converting a string into integer, not ParseInt or parseint

• In C the problem can be avoided by exlusive use of lowercase letters for names

• In Java and C#, many of the predefined names include both uppercase and lowercase letters, so the problem cannot be escaped

• In Fortran 90, lowercase letters are allowed, and they simply translated to uppercase letters

Special words:

- reserved words (special words that cannot be used as names):
- Can't define for or while as function or variable names.
- Good design choice

Keywords (special only in certain context):

• In FORTRAN, if REAL is in the beginning of a statement and followed by a name, it is considered as a keyword for declaration.

Examples

REAL APPLE declaration

REAL = 8.7 assignment

or

INTEGER REAL

REAL INTEGER

This is allowed but not readable.

• Special words and names are distinguished by content

Special words :

Predefined names (have predefined meanings, but can be redefined by the user):

For example, built-in data type names in Pascal, such as INTEGER, normal input/output subprogram names, such as readln, writeln, are predefined.

2.2 VARIABLES

In programming, a variable is a value that can change, depending on conditions or on information passed to the program. Typically, a program consists of instruction s that tell the computer what to do and data that the program uses when it is running. The data consists of constants or fixed values that never change and variable values (which are usually initialized to "0" or some default value because the actual values will be supplied by a program's user). Usually, both constants and variables are defined as certain data types. Each data type prescribes and limits the form of the data. Examples of data types include: an integer expressed as a decimal number, or a string of text characters, usually limited in length.

In object-oriented programming, each object contains the data variables of the class it is an instance of. The object's method s are designed to handle the actual values that are supplied to the object when the object is being used. Abstraction of a computer memory cell or collection of cells

- It is not just a name for a memory location
- It has six attributes: name, address, value, type, lifetime, Scope

1. Name

• Most variables are named (often referred as identifiers).

• Although nameless variables do exist (e.g. pointed variables).

2. Address

• Associated memory location

• It is possible that the same name refer to different locations

1. in different parts of a program.

– A program can have two subprograms sub1 and sub2 each of defines a local variable that use the same name.

e.g. sum

2. in different times.

– For a variable declared in a recursive procedure, in different steps of recursion it refers to different locations.

Aliases

- Multiple identifiers reference the same address – more than one variable are used to access the same memory location
- Such identifier names are called aliases.
- Aliases are not good for readability because the value of a variable can be changed by an assignment to its another name.
- can be created explicitly
    - by EQUIVALENCE statement in FORTRAN
    - by union types in C and C++
    - by variant record in Pascal
    - by subprogram parameters
    - by pointer variables

3. Type

• Determines the range of values the variable can take, and the set of operators that are defined for values of this type.

• For example int type in Java specifies a range of -2147483648 to 2147483647

4. Value

- Contents of the memory cell or cells associated with the variable
- (abstract memory cell instead of byte size memory cells)
- l_value ← r_value (assingment operation)
- l_value of a variable: address of the variable
- r_value of a variable: value of the variable