**WHILE**

**WHILE statement**

WHILE repeats a block of statements as long as a specific condition or set of conditions is true.

ÊÊ WHILE logical expression ; ÊÍ

ÊÊ »

statement ;

ÊÍ

ÊÊ END ; ÊÍ

**logical expression**

1. logical expression

(logical expression) condition

AND

OR

logical expression

ÊÍ

**condition**

1. VisualAge Generator: Programmer's Reference

ÊÊ

|  |  |  |  |
| --- | --- | --- | --- |
| map item |  |  | IS |
|  |  |
|  |  |  | NOT |
|  |  |  |

SQL row record item

BLANK

BLANKS

CURSOR

DATA

MODIFIED

NULL

NULLS

NUMERIC

IS

NOT

BLANK

BLANKS

NULL

NUMERIC

TRUNC

**WHILE**

ÊÍ

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| record | | |  |  |  |  |  | IS | |  |  |  |  | I/O error value | | |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | NOT | | |  |  |  |  |  |  |
| map |  |  |  | IS | | |  |  |  | MODIFIED | |  |  |
|  |  |  |  | |  | | |  |  |  |
|  |  |  |  | |  | | |  |  |  |
|  |  |  |  |  | NOT | | |  |  | | |  |  |  |  |  |
|  |  |  |  |  |  |  | | |  |  |  |  |  |
| EZEAID | | |  |  |  |  |  | IS | |  | |  |  | AID value | |  |
|  |  |  |  |  |  | |  |  |  |
|  |  |  |  |  |  |  |  | NOT | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| data item | | |  |  |  |  | IS | | | |  |  |  |  |  |  |  | BLANK | |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | NOT | | | |  |  |  |  |  |  | BLANKS | | |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | NUMERIC | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| data item | | |  |  |  |  | EQ | | | |  | |  |  |  | data item | | | | |  | | |
|  |  |  |  |  | |  |  |  |  | | |
|  |  |  |  |  |  |  | = | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | NE | | | |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | Ï= | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | =Ï | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |= | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | =| | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | GT | | | |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | > | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | LT | | | |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | < | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | GE | | | |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | >= | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | => | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | LE | | | |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | <= | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | =< | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | IN | | |  | |  |  |  |  |  |  |  |  |  |  |  |
| EZESYS |  |  |  | IS | |  |  |  | |  |  | SYS value | | | |  |  |  |  |  |
|  |  | |  | | | |  | | | |  |  | | | | |
|  |  | |  | | | |  | | | |  |  | | | | |
|  |  |  |  | NOT | | | |  | |  | | | |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | |  | | | |  |  |  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| **Attribute** | **Description** |
|  |  |
| statement | Any statement or the line that represents the execution of the I/O |
|  | option within statement definition. |
|  |  |
| AND, OR | Connectors that can be used to test multiple conditions. With AND, |
|  | both conditions must be met. With OR, either condition can be met. A |
|  | combination of AND and OR can be used, but AND is evaluated |
|  | before OR unless you use parentheses to control the order. Multiple |
|  | ANDs and ORs can be specified on a single line. |
|  |  |

Chapter 10. Program processing statements **435**

**WHILE**

|  |  |
| --- | --- |
| **Attribute** | **Description** |
|  |  |
| map item | Name of a variable field on a map or a map item parameter for a |
|  | function. A map item can be subscripted, qualified, or both. This |
|  | comparison is only valid for terminal maps. |
|  |  |
| IS | Boolean operator that tests true if the specified state is true. |
|  |  |
| NOT | Boolean operator that tests true if the specified state is false. |
|  |  |
| BLANK, | When used with map items, tests true if either of the following cases |
| BLANKS | are true: |
|  | v The data received from the display for the specified data item |
|  | contained all blanks or nulls or both. |
|  | v The map containing the item has not been conversed since the |
|  | program started, or since the last SET map CLEAR. |
|  | When use with non-map items with data type CHA, MIX, or DBCS, |
|  | tests true if the data item contains all blanks. |
|  |  |
| CURSOR | Tests that the user left the cursor in the specified data item. |
|  |  |
| DATA | Tests that there is data other than blanks or nulls within the map item |
|  | specified. Either the user entered the data or the data was moved to |
|  | the field before writing to the screen. |
|  |  |
| MODIFIED | Tests true if data in the variable field has changed. Data is considered |
|  | changed if any of the following conditions are true: |
|  | v When specified for a map variable field, data was entered by the |
|  | program user the last time the map was displayed. |
|  | v A SET MODIFIED was done prior to the CONVERSE of the map. |
|  | v The field on the map was defined with a modified data tag (MDT) |
|  | at map definition time, and this is the first display of the map in the |
|  | program or the first display of the map after a SET CLEAR. |
|  | v When specified for a map, tests true if any variable field on the map |
|  | was changed. |
|  | **Note:** This saves you from having to test each map field separately. |
|  |  |
| NULL, | When specified for map variable field, tests true if either of the |
| NULLS | following cases are true: |
|  | v The data entered into the panel for the specified data item contained |
|  | all nulls or blanks. Nulls are received when the program user |
|  | presses the Erase EOF key. Note that a true TEST for NULLS does |
|  | not mean that the field contains nulls internally (it contains blanks). |
|  | v The map containing the item has not been conversed since the |
|  | program started, or since the last SET map CLEAR. |
|  | When use with non-map items with data type CHA, MIX, DBCS, or |
|  | UNICODE, tests true if the data item contains all blanks. |
|  |  |

1. VisualAge Generator: Programmer's Reference

|  |  |
| --- | --- |
|  | **WHILE** |
|  |  |
| **Attribute** | **Description** |
|  |  |
| NULL (SQL | Tests true if the SQL row record item has had no value assigned to the |
| row record | item. |
| item) |  |
|  |  |
| NUMERIC | If the map item or data item type is character or mixed, tests true if |
|  | the field contains the characters 0 through 9. NUMERIC cannot be |
|  | used with EZE words. |
|  |  |
| SQL row | Name of a data item in an SQL row record or an SQL item parameter |
| record item | for a function. The name can include a qualifier. |
|  |  |
| TRUNC | Tests whether a character or a DBCS item in an SQL row record was |
|  | truncated (nonblank characters deleted on the right) the last time the |
|  | item value was read from the relational database. Truncation can only |
|  | occur when the column in the database is longer than the data item. |
|  | The TRUNC indicator is reset whenever a value is moved to the item, |
|  | or when the item is set to NULL. |
|  |  |
| record | Name of a record. |
|  |  |
| I/O error | Tests true if the I/O error value specified was returned from the |
| value | system on the last I/O option that accessed the record. See [ªI/O error](#page396) |
|  | [valueº on page 368](#page396) for more information. |
|  |  |
| map | Name of a map. |
|  |  |
| EZEAID | The special function used to test the key that caused the input |
|  | interrupt from the display. |
|  |  |
| AID value | Used in testing the state of the EZEAID special function word. See |
|  | [ªAID valueº on page 364](#page392) for more information. |
|  |  |
| data item | A data item syntactical element. See [ªData itemº on page 366](#page394) for more |
|  | information. |
|  |  |
| EQ or = | Boolean operators that test true if data item values are equal. |
|  |  |
| NE, Ï=, =Ï, | Boolean operators that test true if data item values are not equal. |
| |=, or =| | **Note:** The Ï and | symbols are not in the national language syntactic |
|  | character set, and might not have an equivalent code point across |
|  | different code pages. If you are exporting your program or generating |
|  | for machines with differing code pages (in particular, between |
|  | System/370 host systems and workstations), use NE, not the symbols. |
|  |  |
| GT or > | Boolean operators that test true if the value of the first data item is |
|  | greater than the second. |
|  |  |
| LT or < | Boolean operators that test true if the value of the first data item is less |
|  | than the second. |
|  |  |
| GE, >=, or => | Boolean operators that test true if the value of the first data item is |
|  | greater than or equal to the second. |
|  |  |
| LE, <=, or =< | Boolean operators that test true if the value of the first data item is less |
|  | than or equal to the second. |
|  |  |

Chapter 10. Program processing statements **437**

**WHILE**

|  |  |
| --- | --- |
| **Attribute** | **Description** |
|  |  |
| IN | Boolean operator that tests true if the value in the first data item can |
|  | be found in the array represented by the second data item. |
|  | If a match is not found, processing skips to the corresponding END |
|  | statement. |
|  | **Note:** The value of special function word EZETST is set to 0 if a match |
|  | is not found. If a match is found, EZETST is set to the index number |
|  | of the first element of the array that matches the value of the data |
|  | item. |
|  | Successive items in the array are compared until a match is found or |
|  | the end of the array is reached. If the array includes an index, the |
|  | testing starts there rather than from the first item in the array. If no |
|  | starting index is given, the test starts with the first item in the array. If |
|  | the value of the starting index is greater than the number of entries in |
|  | the array or if no match is found, the test will test false. |
|  | Comparing against a single data item instead of an array is equivalent |
|  | to comparing for equal, but is slower and causes setting of EZETST to |
|  | 0 or 1. It will not be treated as an error. |
|  | The IN function is similar to the FIND statement in that they both scan |
|  | for values, but you would use IF or WHILE rather than FIND in the |
|  | following situations: |
|  | v IN works with any array, not just a table column. |
|  | v The search does not have to start at the first entry of the array. |
|  | v Duplicate values can be found in the array. |
| EZESYS | The EZESYS test is a runtime test. Generation for a target system will |
|  | fail if the program includes functions not supported on that system, |
|  | even if the function is within an IF EZESYS clause that would prevent |
|  | that function from executing on the target system. To enable |
|  | generation for the target system to proceed, replace the offending |
|  | function with a call to a program that performs the function. |
|  |  |
| SYS value | Used to test the state of the EZESYS special function word. See [ªSYS](#page405) |
|  | [valueº on page 377](#page405) for more information. |
|  |  |

**Uses**

Parentheses can be used to control how conditions are evaluated.

When a conditional expression is nested within parentheses, evaluation proceeds from the least inclusive to the most inclusive part of the expression. The nested expression is evaluated before the expression which contains it. Unless the evaluation order is modified by parentheses, the AND operator is evaluated before the OR operator.

Parentheses can be used to:

1. VisualAge Generator: Programmer's Reference

**WHILE**

1. Modify the normal Boolean precedence of operations
2. Eliminate ambiguities where operations appear at the same level.

This block of statements controlled by a conditional statement can contain conditional statements. This can continue to a maximum of 15 levels deep.

When WHILE is used in the flow stage of a function and a function name is specified in the block of statements controlled by the WHILE, control is passed to the function named and does not return.

When using the WHILE statement within a function, you may test the map attributes of a parameter item as long as the parameter item has been defined as a map item parameter or the SQL attributes as long as the parameter item has been defined as an SQL item parameter. This capability allows reusable routines to be written to handle the map and SQL item processing.

**Target environments for WHILE**

ASCII character sets are used in workstation environments. EBCDIC character sets are used in host environments. Differences in collating sequence can cause greater-than or less-than comparisons to have different results in ASCII environments than in EBCDIC environments.

|  |  |
| --- | --- |
| **Environment** | **Compatibility considerations** |
|  |  |
| VM CMS | Uses EBCDIC character sets. |
|  |  |
| VM batch | Uses EBCDIC character sets. |
|  |  |
| CICS for | Uses EBCDIC character sets. |
| MVS/ESA |  |
|  |  |
| MVS/TSO | Uses EBCDIC character sets. |
|  |  |
| MVS batch | Uses EBCDIC character sets. |
|  |  |
| IMS/VS | For a map field to test true when the data entered for a data item |
|  | contained all blanks, nulls, or a combination of both, the program |
|  | user must enter at least one blank in the field before pressing the |
|  | Erase EOF key. If the program user presses Erase EOF without |
|  | entering one blank in the field, IMS message format services leave |
|  | Uses EBCDIC character sets. |
|  |  |
| IMS BMP | Uses EBCDIC character sets. |
|  |  |
| CICS for | Uses EBCDIC character sets. |
| VSE/ESA |  |
|  |  |
| VSE batch | Uses EBCDIC character sets. |
|  |  |
| CICS for OS/2 | Range check comparisons for character data are performed using the |
|  | ASCII collating sequence. |
|  |  |
| OS/400 | Uses EBCDIC character sets. |
|  |  |

Chapter 10. Program processing statements **439**

**WHILE**

|  |  |  |
| --- | --- | --- |
| **Environment** | **Compatibility considerations** | |
|  |  | |
| OS/2 (GUI) | Range check comparisons for character data are performed using the | |
|  | ASCII collating sequence. | |
|  | The following are not supported: | |
|  | v | WHILE record IS I/O error value |
|  | v | WHILE record NOT I/O value |
| Windows (GUI) | Same as OS/2 (GUI). | |
|  |  | |
| OS/2 (C++) | Uses ASCII character sets. | |
|  |  | |
| AIX | Uses ASCII character sets. | |
|  |  | |
| HP-UX | Uses ASCII character sets. | |
|  |  | |
| CICS for AIX | Uses ASCII character sets. | |
|  |  | |
| Windows NT | Uses ASCII character sets. | |
| (C++) |  |  |
|  |  | |
| Windows NT | Uses ASCII character sets. | |
| (Java) |  |  |
|  |  | |
| CICS for | Uses ASCII character sets. | |
| Windows NT |  |  |
|  |  | |
| Solaris | Uses ASCII character sets. | |
|  |  | |
| CICS for Solaris | Uses ASCII character sets. | |
|  |  | |
| Test Facility | No distinction is made between testing for BLANKS and NULLS. | |
|  | Uses ASCII character sets. | |
|  |  |  |

**Examples for WHILE**

The following examples show you how to use the WHILE statement:

**WHILE statement**

The following is an example of a WHILE statement:

WHILE NUMRECS > 0 AND

(REQTYPE = 1 OR

REQTYPE = 2);

/\* While there is more data and /\* request type is 1 or /\* request type is 2.

**WHILE statement summing the elements in an array**

The following statements sum the elements in an array, which has 50 entries, but stop early if an entry greater than 9999 is found:

MOVE 0 TO TOTAL;

MOVE 1 TO J;

WHILE J LE 50

AND ARRAY[J] LE 9999;

TOTAL = TOTAL + ARRAY[J];

J=J+1;

END;

1. VisualAge Generator: Programmer's Reference

**WHILE**

The two conditions being tested are that J is less than or equal to 50 (that is, the subscript is within the range of the array) and that the entry is less than or equal to 9999. As long as both conditions are met (AND), the entry is added to the sum of the previous entries (TOTAL) and the subscript (J) is incremented. When either condition is not met, the summation stops.

**WHILE statement using the IN operation**

The IN operation provides a function similar to FIND, but much more powerful. One data item (called ITEM in the example below) is compared to another data item (called LIST in the example below) to ensure they match. The LIST data item must be an array. Successive items in the array are compared until a match is found or the end of the array is reached. If LIST includes an index, the search begins there rather than at the first item in the array. See the following example:

|  |  |  |  |
| --- | --- | --- | --- |
| MOVE 1 TO START; | /\* | Beginning | of array |
| NUMBER-OF-OCCURS = 0; | /\* | None found yet | |
| WHILE ITEM IN LIST[START]; | /\* | Find next | occurrence |
| NUMBER-OF-OCCURS = NUMBER-OF-OCCURS + 1; /\* | | Count number of occurrences | |
| START = EZETST + 1; | /\* Skip past | | the one found |
| END: |  |  |  |

At this point NUMBER-OF-OCCURS contains the number of times the value in ITEM appears in the array LIST.

The above example is intended to determine the number times a value occurs in an array. The loop will terminate if any of the following conditions occur:

1. If ITEM does not appear in LIST
2. When START becomes greater than the number of entries in the array LIST.
3. When there are no occurrences of the value of ITEM in LIST at or above the entry previously found.

If no starting index is specified, the test starts with the first item in the array. If the value of the starting index is greater than the number entries in the array, the test will test false. It will not be treated as an error.

The following are some advantages of using IN instead of the FIND statement:

1. IN works with any array, not just a table column.
2. The search does not have to begin at the first array entry.
3. Duplicate values can be found in the array.

The following table shows which data items can be compared with each other:

Chapter 10. Program processing statements **441**

**WHILE**

*Table 19. Valid data item comparisons*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  | **BIN** | **CHA** | **DBCS** | **HEX** | **MIX** | **NUM** | **NUMC** | **PACK** | **PACF** | **UNICODE** | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| BIN | x |  |  |  |  | x | x | x | x |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| CHA |  | 1 |  | 2 | 1 | 3 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| DBCS |  |  | 1 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| HEX |  | 2 |  | 4 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| MIX |  | 1 |  |  | 1 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| NUM | x | 3 |  |  |  | x | x | x | x |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| NUMC | x |  |  |  |  | x | x | x | x |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| PACF | x |  |  |  |  | x | x | x | x |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| PACK | x |  |  |  |  | x | x | x | x |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |
| UNICODE |  |  |  |  |  |  |  |  |  | 1 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

**Legend:**

1. Valid data item comparison
2. For CHA to CHA, DBCS to DBCS, MIX to MIX, UNICODE to UNICODE, CHA to MIX, or MIX to CHA comparisons,the shorter item is logically padded on the right with blanks to the length of the longer item. All comparisons are logical comparisons.
3. Valid only if CHA field contains hexadecimal characters (a-f, A-F, 0-9). If a HEX item is compared to a CHA item, the CHA item is converted to HEX format, the shorter field is padded on the right with binary zeros, and a logical comparison is made.
4. Indicates that the data content of the source is validated prior to comparison. If nonnumeric, the program is abnormally terminated. Valid only if the numeric field is defined without decimal positions. The shorter field is padded on the left with zeros.
5. If a HEX item is compared to a HEX item, the shorter field is padded on the right with binary zeros to the length of the longer field, and a logical comparison is made.