Esoteric four-character programming language

**HELLSMI**

Programming reference

**1**

**Description**

The esoteric programming language “HELLSMI” is an intentionally limited symbolic language that uses only the basic symbols used in chat communication to express two basic emotions: joy ( '**)**’ и '**0**' ) and sadness ( '**(**' и '**9**' ). The name of the language is formed from the words “Hellish” and “Smile”. Since programming in it requires a certain perseverance and the source code is a bit like expressing emotions in chat or other means of exchanging messages.

**Syntax**

Any program source code begins with the label "**()90**". Further, without any separator, follows the number of necessary numeric memory blocks for the program to work in a special record. As an example:

**((9(9)9)9)0)**

In this case, 21000 memory blocks (i.e. from 0 to 20999) are allocated for work for storing and exchanging numerical information in them. The number of brackets is equal to a digit, zero is indicated by an inverse bracket, the symbol ‘**9**’ moves to the least digit and the label “**0)**” signals the end of the line. If we need a negative number, then "**)0**" is added to the beginning - record the minus sign. All numeric strings in the language are formatted in this form, therefore, for convenience, the set includes a coder program (**hncode**) that translates from a regular number into a HELLSMI record and vice versa.

After the declaration of the amount of memory, a transition to another line is carried out, from this moment operators begin, each of which also ends with a transfer to another line (except for the last). All commands have a standardized syntax:

**XXXX{First parameter}…{Х parameter}**

Where XXXX is the command number according to the list of available commands (on page 3). This is followed by the numeric parameter strings formatted with the above entry. For example, the command to write to a block of memory:

**0009)0)(9(((0)**

The command ("**0009**") writes to cell 0 ("**)0)**") the number 13 ("**(9(((0)**").

**General information**

Language class: Esoteric

Language type: Interpreted (possibly compiled in the future)

Number of characters available: 4 (‘**9**’, ‘**0**’, ‘**(**’, ‘**)**’)

Number of operators available: 24

Memory block type: Integer

Working with characters: By converting decimal ASCII codes

**2**

**List of available commands**

* **0009 – Writing a numeric value to the specified block**

0009{Block number}{Numerical value}

* **9000 – Reading a numeric value from a block and displaying it**

9000{Block number}

* **9((9 – Display a numerical sequence (Separator - space)**

9((9{From block}{To block}

* **9999 –** **Increase the block value by some number**

9999{Block number}{Adding number}

* **0000 – Decrease block value by some number**

0000{Block number}{Subtracting number}

* **0099 – Add two blocks and write the result into another block**

0099{Result block}{First adding block}{Second adding block}

* **0900 – Subtract one block from another and write into another block**

0900{Result block}{Reducing block}{subtracting block}

* **0909 – Multiply two blocks and write into another block**

0909{Result block}{First multiplier block}{Second multiplier block}

* **0990 – Split one block entirely into another and write into another block**

0990{Result block}{Divisible block}{Divider block}

* **0999 – Split one block into another with the remainder write into another block**

0999{Result block}{Divisible block}{Divider block}

* **9009 – Display an ASCII character by decimal code from a block**

9009{Block number}

* **9))9 – Display a sequence of ASCII characters by decimal codes from blocks**

9))9{From block}{To block}

* **9090 – Copy the value of one block to another**

9090{Destination block}{Source block}

* **9099 – Resetting the program (return to the second line)**

9099

* **9900 – WHILE loop**

9900{First condition block}{Second condition block}(){Comparison type (Page 5)}

**3**

* **9909 – IF condition**

9909{First condition block}{Second condition block}(){Comparison type (Page 5)}

* **9990 – Jump to the specified line of the program**

9990{Program line number}

* **0()0 – Entering a numeric value into a block from the keyboard („?>”)**

0()0{Block number}

* **0)(0 – Entering an ASCII character (as a code) into a block from the keyboard („A?>”)**

0)(0{Block number}

* **9()9 – Entering a numerical sequence into blocks from the keyboard („S?>”) (Separator - space)**

9()9{From block}{To block}

* **9)(9 – Entering a sequence of ASCII characters into blocks from the keyboard („SA?>”) (By block per character, separator - space)**

9)(9{From block}{To block}

* **(09) – End of the body of the WHILE loop**

(09)

* **(90) – End of IF condition body**

(90)

* **0090 – Exiting the program**

0090

**4**

**Comparison Operators**

* **()9 – If equal**

If the values of the blocks are equal, the body will be executed

* **()0 – If not equal**
* If the values of the blocks are not equal, the body will be executed
* **()( – If the first is less than the second**

If the value of the second block exceeds the first, the body will be executed

* **()) – If the first is greater than the second**

If the value of the first block exceeds the second, the body will be executed

* **()(( – If the first is less than or equal to the second**

In case of superiority or equality of the value of the second block over the first, the body will be executed

* **())) – If the first is greater than or equal to the second**

In case of superiority or equality of the value of the first block over the second, the body will be executed

**5**