**Mini-Language Specification**

Alphabet

* A-Z, a-z (Uppercase and lowercase letters)
* 0-9 (Digits)
* \_

Operators

* +, -, \*, /, % (Addition, Subtraction, Multiplication, Division, … - Arithmetic)
* ==, !=, >, < (Equality, Inequality, Greater, Equal - Relational)
* &&, ||, ! (And, Or, Not - Logical)
* = (Assignment)

Separators (for this document’s readability each separator will be in ““)

* “;”, “,”, “ “, “{“, “}”, “(“, “)”

Keywords

* read
* write
* if
* else
* for
* while
* break
* integer
* string
* character
* array
* return

Identifiers

* identifier = letter {letter|digit} | letter
* letter = “A”| “B”| …| “Z”| “a”| “b”| …|”z”
* digit = “0”| non\_zero\_digit
* non\_zero\_digit = “1”| … “9”

Constants

* integer = “0” |[“+”|”-“] non\_zero\_digit {digit}
* character = ‘letter’|’digit’
* string = “{letter|digit}”

|  |
| --- |
| **Token** |
| [ |
| ] |
| { |
| } |
| ; |
| : |
| , |
| < |
| > |
| == |
| != |
| ! |
| && |
| || |
| = |
| + |
| - |
| \* |
| / |
| % |
| character |
| integer |
| string |
| array |
| if |
| else |
| for |
| while |
| break |
| return |
| read |
| write |
| start |

Syntax

* program = “start” compound\_statement
* declaration = type “ “ identifier
* simple\_type = “integer” | “string” | “character”
* array\_declaration = simple\_type “array” “[“ integer “]”
* type = simple\_type | array\_declaration
* compound\_statement = “{“ statement\_list “}”
* statement\_list = statement| statement “;” statement\_list
* statement = simple\_statement | struct\_statement
* simple\_statement = assign\_statement | io\_statement | declaration
* struct\_statement = compound\_statement | if\_statement | while\_statement | for\_statement
* if\_statement = “if” condition statement [“else” statement]
* for\_statement = “for” “(“ “number” assign\_statement “;” condition “;” assign\_statement “)” statement
* while\_statement = “while” condition statement
* assign\_statement = Identifier “=“ expression
* expression = [expression (“+”|”-”)] term
* term = term (“\*” | “/”) factor | factor
* factor = “(“ expression “)” | integer | Identifier| Identifier “[“ integer “]”
* io\_statement = (“read” IDENTIFIER) | (“write” (Identifier| Constant))
* condition = “(“ expression relation expression “)”
* relation = “<“ | “<=“ | “==“ | “!=“ | “>=“ | “>“

Examples

P1.

Requirement: Compute the maximum number out of 3 input numbers and display it.

Solution:

start {

integer a;

integer b;

integer c;

integer max;

read a;

read b;

read c;

if(a>b&&a>c){

max=a;

}

else{

if(b>c&&b>a){

max=b;

}

else{

max=c;

}

}

write max;

}

P2.

Requirement: Check if an input is a prime number.

Solution:

start{

integer a;

integer i;

integer is\_prime;

is\_prime=0;

read a;

for(i=2;i<a;i=i+1){

if(a%i==0){

is\_prime=1;

break;

}

}

if(is\_prime==1){

write "a is prime"

}else{

write "a is not prime";

}

}

P3.

Requirement: Compute the sum of n input numbers which are bigger than m (another input number) and display it.

Solution:

start{

integer n;

integer m;

integer sum;

integer current\_number;

sum=0;

read n;

read m;

for(i=0;i<n;i=i+1){

read current\_number;

if(current\_number>m){

sum=sum+current\_number;

}

}

write sum;

}

P1err.

Requirement: Compute the sum of 2 input numbers and display it.

Solution:

start{

integer 2a; <- lexical error

integer b;

integer sum;

sum=0;

read a;

read b;

sum+=b; <- lexical error

sum=sum+a;

write sum;}