ARIZONA STATE UNIVERSITY

The G28 Programming Language

A Reference Manual

Contents

1. Introduction ..................................................................................................................................... 3

2. Compiling and Running G28 programs ....................................................................... 3

3. G28 Operators ………................................................................................................................... 5

4. Sample program in G28 ......................................................................................................... 6

5. G28 Type System ………………............................................................................................... 7

6. Declarations and Arithmetic ............................................................................................ 7

7. The if-else condition ................................................................................................................ 8

8. The while condition................................................................................................................... 9

9. I/O Operations……………………............................................................................................... 9

10. Functions ….................................................................................................................................. 10

11. The Factorial program without recursion ............................................................... 11

Introduction

The design goal of G28 is to create a new language in order to understand the design and implementation issues of designing a new implementation language.

Tools used:

* Flex is used as a scanner/lexer
* Bison is used for parsing

The G28 high level language is both compiled and interpreted. Upon compilation of the flex and bison files, an executable is generated which is in turn used to generate the intermediate files. The G28 executable compiles a source program into an intermediate file while the G28 runtime reads the intermediate file line by line and executes it.

The runtime for the G28 language is written in C Sharp.

Compiling and Running G28 programs

G28 source files should end in .g28 extension and the intermediate code generated has a .int extension.

Steps to generate the executable:

* flex g28.l
* bison -d G28.y
* gcc -o G28.g28 lex.yy.c G28.tab.c -ll -lm

To generate the intermediate code using the executable, run this command:

./G28.g28 filename.g28

To compile the runtime and execute the intermediate code, run these commands:

C#: [on mac OSX/LINUX]

1. mcs G28Run.cs
2. mono G28Run.exe filename.g28.int

Steps to install C# on mac OSX:

Download here [<http://www.mono-project.com/download/>]

Follow guidelines here [<http://www.mono-project.com/docs/about-mono/supported-platforms/osx/>]

Steps to install C# on LINUX:

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF

echo "deb <http://download.mono-project.com/repo/debian> wheezy-apache24-compat main" | sudo tee -a /etc/apt/sources.list.d/mono-xamarin.list

sudo apt-get update

sudo apt-get install mono-mcs

G28 Operators

“+” Addition

“-” Subtraction

“\*” Multiplication

“/” Division

“EQ” Assignment

“CON” If-condition

“INSTEAD” Else

“T” True(Boolean)

“F” False(Boolean)

“&” And operator

“|” Or Operator

“()”,”{}”,”[]” Parenthesis

“LOOP UNTIL” While condition

“STK” Stack declaration

“FUNC” Function declaration

“LFT”, “RHT” Beginning and end of function

“CALL” Function calling

“SCAN” Read input

“FLASH” Print

“:” End of each statement

“PUSH” Push operation on stack

“POP” Pop operation on stack

“PEEK” Peek operation on stack

“GT” Greater than

“LS” Lesser than

Sample program in G28

SCAN a:

LOOP UNTIL ( a GT 1) { :

FLASH a:

a EQ a -1:

}:

The above program takes an input ‘a’ from user and prints it in a loop by decrementing the integer each time using a while loop.

Output:

Enter a value:

5

5.0

4.0

3.0

2.0

The G28 type system

G28 supports the following types:

* int – for signed integer types
* boolean – for boolean variables

In the code mentioned below, we define variables of the two types and print their values to the standard output:

a EQ 22 :

b EQ T :

FLASH a:

FLASH b:

Output:

22.0

TRUE

Declarations and Arithmetic

The following code demonstrates the declarations and arithmetic operations:

SCAN a :

SCAN b :

c EQ a + b :

FLASH c :

c EQ a – b:

FLASH c :

c EQ a \* b:

FLASH c :

c EQ a / b :

FLASH c :

Output:

Enter a value:

4

2

6.0

2.0

8.0

2.0

The if-else condition

G28 language supports the if-else conditional statement. Below is an example:

SCAN a :

product EQ 1 :

CON ( a GT 5) [ :

product EQ a \* 5 :

] :

INSTEAD [ :

product EQ a\*4 :

] :

FLASH a :

Output:

Enter a value:

6

30.0

The while condition

G28 language supports the while condition as well.

SCAN a:

LOOP UNTIL ( a GT 1) { :

FLASH a:

a EQ a -1:

}:

The above program takes an input ‘a’ from user and prints it in a loop by decrementing the integer each time using a while loop.

Output:

Enter a value:

5

5.0

4.0

3.0

2.0

I/O Operations

G28 supports reading the input from the user using the SCAN operation. The output is provided to the user using the FLASH operation.

The example for I/O operation is illustrated below:

a EQ 17 :

b EQ F :

FLASH a:

FLASH b:

Output:

17.0

FALSE

Functions

The rules of using functions in G28 language are as follows:

* They are defined and not declared. Later they can be called.
* They do not return a value.

The following is an example:

a EQ 5

FUNC abc LFT :

CON ( a GT 1 ) [ :

a EQ 5 \* a :

a EQ a - 1 :

CALL abc :

] :

RHT :

CALL abc :

FLASH a :

Output:

24

Factorial program without recursion

SCAN a :

product EQ 1 :

sum EQ 0 :

FUNC abc LFT :

CON ( a GT 1 ) [ :

product EQ product \* a :

a EQ a - 1 :

CALL abc :

] :

RHT :

CALL abc :

FLASH product :

Output:

Enter a value:

4

24.0