Introduction to informatics

Piroska Biró

Course 09

Exercises

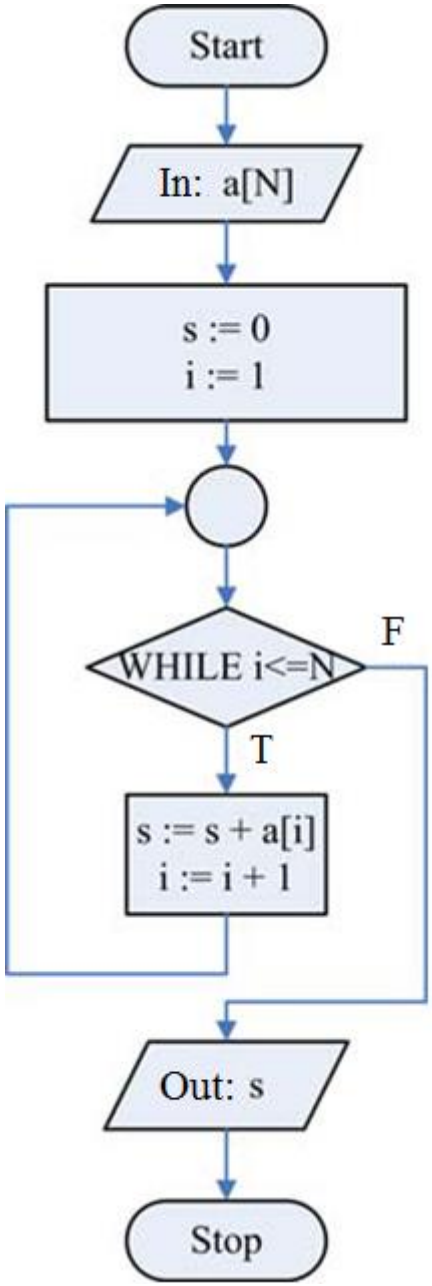
Write the algorithm of the following exercices using the following language of description:

Pseudocode, Flow chart, Structogram

- The sum of N numbers
- The average of N numbers
- From N numbers the geometric average of the positive numbers

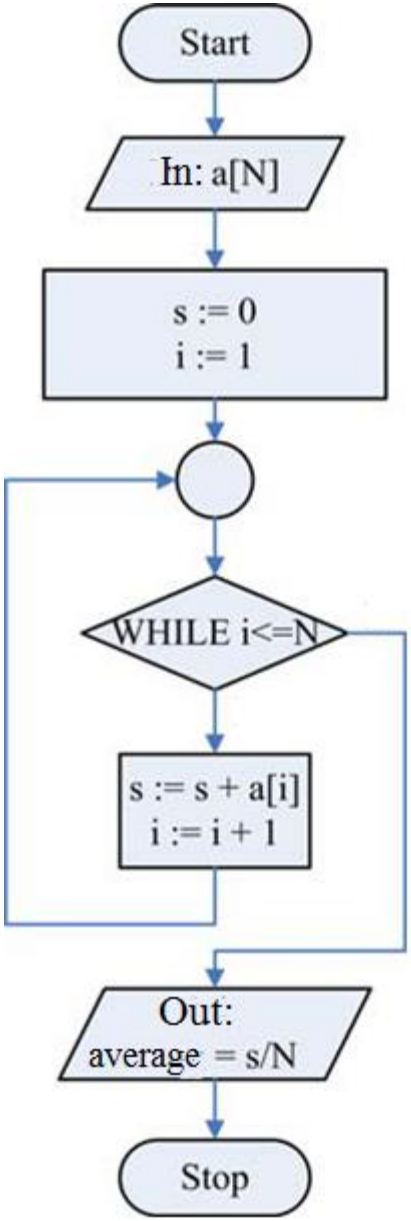
The sum of N numbers

```
BEGIN
        INPUT: a[N]
        i := 1, s := 0
        WHILE i <= N
                s := s + a[i]
                i := i + 1
        END
        OUTPUT: s
END
```



The average of N numbers

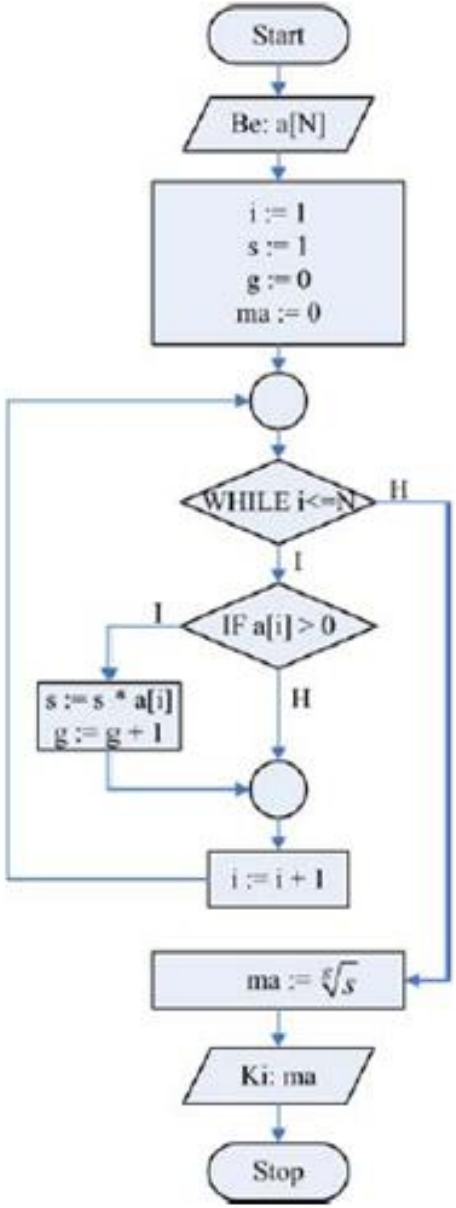
```
BEGIN
        INPUT: a[N]
        i := 1, s := 0
        WHILE i <= N
                s := s + a[i]
                i := i + 1
        END
        OUTPUT: s/N
END
```



From N numbers the geometric average of the positive numbers

$$\bar{x}_g = \sqrt[n]{x_1 * x_2 * \ldots * x_n}$$

```
BEGIN
        INPUT: a[N]
        i := 1, s := 1, g := 0, ma := 0
        WHILE i<=N
                IF a[i] >0
                        s := s*a[i]
                        g := g+1
                END
                i := i+1
        END
        ma := s^(1/g)
        OUTPUT: ma
END
```



C programing language

- developed by Dennis Ritchie between 1969 and 1973 at Bell Labs
- system software like the Unix computer operating system
- the following languages are build on C directly or indirectly: C#, D, Go, Java, JavaScript, Limbo, LPC, Perl, PHP, Python, and Unix's C shell
- book by Ritchie and Brian Kernighan; that version is generally referred to as "K&R" C
- American National Standards Institute published a standard for C ("ANSI C" or "C89")

Book

- 1978, Brian Kernighan and Dennis Ritchie published the first edition of „The C Programming Language”
- C programmers as „K&R”
- K&R introduced several language features:
  - standard I/O library
  - long int data type
  - unsigned int data type
  - compound assignment operators of the form

Keywords/Reserved words in C

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Variables

- differences between small and capital letter
- using **&** operator we can refer to the address of (the variable) **a**, **&a**
- **char, int:** store the integer numbers
- **char:** store characters
- **float, double:** store the real numbers

Definition of the variables

- **int** a;
- **float** b1, b2;
- **long** i, j=2, k;
- **unsigned char** c=65;
- **long double** x, y=3.14;

Types of data

char	8	−128 .. 127
unsigned char	8	0 .. 255
short	16	−32768 .. 32767
unsigned short	16	0 .. 65535
int	16	−32768 .. 32767
int	32	−2147483648 .. 2147483647
unsigned int	16	0 .. 65535
unsigned int	32	0 .. 4294967295
long	32	−2147483648 .. 2147483647
unsigned long	32	0 .. 4294967295
float	32	$3.4 \cdot 10^{-38}$ .. $3.4 \cdot 10^{38}$
double	64	$1.7 \cdot 10^{-308}$ .. $1.7 \cdot 10^{308}$
long double	80	$3.4 \cdot 10^{-4932}$ .. $1.1 \cdot 10^{4932}$

Format specifiers

Type	Format specifiers
char	%c
int	%d or %i (10-es), %o (base 8), %x, %X (base 16)
unsigned int	%u
short int	%hd or %hi
unsigned short int	%hu
long int	%ld or %li
unsigned long int	%lu
float	%f
double	%lf
long double	%Lf
karakterlánc	%s

Arithmetic operators

Basic assignment		a=b
Addition		a+b
Subtraction		a-b
Unary plus		+a
Unary minus		-a
Multiplication		a*b
Division		a/b
Modulo (integer remainder)		a%b
Increment	prefix	++a
	suffix	a++
Decrement	prefix	--a
	suffix	a--

Comparison operators (relational operators)

Equal to	a==b
Not equal to	a!=b
Greater than	a>b
Less than	a<b
Greater than or equal to	a>=b
Less than or equal to	a<=b

```
a == 5
    /* Does NOT assign five to a. Rather, it checks to see if a equals 5.*/
```

Logical operators

Logical negation (NOT)	!a
Logical AND	a&&b
Logical OR	a||b

Example:

!5, !!5, 5&&6, 0&&13, 0||12;
0 and 1 logical value!!!

Bitwise operators

Bitwise NOT	~a
Bitwise AND	a&b
Bitwise OR	a|b
Bitwise XOR	a^b
Bitwise left shift	a<<b
Bitwise right shift	a>>b

Compound assignment operators

Addition assignment	a += b	a = a + b
Subtraction assignment	a -= b	a = a - b
Multiplication assignment	a *= b	a = a * b
Division assignment	a /= b	a = a / b
Modulo assignment	a %= b	a = a % b
Bitwise AND assignment	a &= b	a = a & b
Bitwise OR assignment	a |= b	a = a | b
Bitwise XOR assignment	a ^= b	a = a ^ b
Bitwise left shift assignment	a <<= b	a = a << b
Bitwise right shift assignment	a >>= b	a = a >> b

Operators

sizeof() operator

- sizeof(a)
- sizeof(type)

Ternary operator

- condition ? value_if_true : value_if_false
- max=a>b **?** a: b

C precedence table

```
( )  []  . ->                                  →
* & + - ! ~ ++ -- SIZEOF (típus)               ←
* / %                                          →
+ -                                            →
>>  <<                                         →
<  >  <=  >=                                   →
== !=                                          →
&                                              →
^                                              →
|                                              →
&&                                             →
||                                             →
?:                                             →
= += -= *= /= %= >>= <<= &= ^= |=              ←
,                                              →
```

Constant in C

- substitute for a sequence of character that cannot be changed, which can be
  - a numeric constant
  - a character constant
  - a string
- #define PI 3.141593
  #define TRUE 1
  #define FALSE 0
- #define PI 3.141593

Commonly used escape sequences

\n	newline
\t	tab
\v	vertical tab
\f	new page
\b	backspace
\r	carriage return
\o	null character
\?	to print question mark
\\	to print slash
\'	to print single quote
\"	to print double quote

Comments

// one line comment

/* */ more line comment

printf()

Syntax

- **printf** (“format string”, argument list);

Example

printf (“Hello world!”);
/*displays the Hello World! text */

printf (”a=%d\nb=%d”,a,b);
/*displays the values of a and b variables*/

scanf()

Syntax

- **scanf** (“Formatted _specifier”, & variable_ name)
- **&** (Address Operator)

Example

```
scanf(”%d”,&a);
scanf(”%d %d”,&a,&b);
scanf(“%d %f”, &i, &j);
```

Statements

Empty statements

Syntax can be of two types:
;
{}

Semantics:
It does not do anything, but we may needed for syntactic purposes.

Statements

Syntax

expression;

Semantics

Execution of the expression.

Examples:

- printf("Hello World!\n");
- x = 2;

IF statement

- if (condition)
  statement;

- if (condition)
  {
  statement 1;
  statement 2;
  }

IF ELSE statement

- if (condition)
  statement1
  else
  {
  statement 2;
  statement 3;
  }

IF-ELSE-IF statement

- if(condition)
  statement 1;
  else if (condition)
  statement 2;
  .....
  .....
  else if(condition)
  statement n-1;
  else
  statemens n ;

Example

```
if (x%2==0)
        {
        printf("x is an even number");
        }
else
      {
      if (x>10)
              {
              printf("x is an odd number and greater than 10");
              }
      else
              {
              printf("x is an odd number and less than 10");
               }
      }
```

Switch statements

- switch (expression)

```
{
 case constant1: statements 1;
 case constant2: statements 2; break;
  ……................
 case constantn-1: statements n-1;
 default: statements n;
}
```