COMPUTER PROGRAMMING

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

DEPARTMENT OF INFORMATION AND COMPUTER ENGINEERING PAPER

2
BASIC ELEMENTS

STUDENT DETAILS:

NAME: ATHANASIOU VASILEIOS EVANGELOS
STUDENT ID: ice19390005
STUDENT STATUS: GRADUATE
PROGRAM OF STUDY: UNIWA
LABORATORY DEPARTMENT: M2
LABORATORY TEACHER: GEORGIOS MELETIOU
DATE TOTAL COMPLETION OF WORK: 9/11/2021

COMPUTER PROGRAMMING

CONTENTS

COMPUTER PROGRAMMING

COMPUTER PROGRAMMING

ANSWERS TO THEORY SUBJECTS

SUBJECT 1

a) <u>What is a variable?</u>

A variable is a memory location in which data is stored, such as, for example, numeric values, characters, strings, arrays, pointers, etc.

b) <u>What are its main characteristics?</u>

The basic characteristics of a variable are as follows:

a) The data is temporarily stored in the memory location.

b) The data entered by the user in a variable can change during the execution of the program.

c) The name of the variable is defined by the user himself, so that he remembers it better.

d) The variable has a data type, such as " int " for integer values, " float " for decimals with few decimal places, " double " for decimals with many decimal places, " char " for characters and " bool " for logicals prices.

SUBJECT 2

a) <u>What is the " standard " input of a program? A program's</u>

" standard " input is an input channel that connects to the corresponding operating system and its derivative programs for data entry. The user enters the data from the keyboard and it is read from the " standard " input.

b) <u>What is the " standard " output of a program?</u>

The " standard " output of a program is an output channel that is also connected to the corresponding operating system and to the derivative programs to print the data. The user

reads the data from the screen and prints it from the " standard " output.

c) <u>What do you know about the functions with which we handle " standard " input in " C " programming?</u>

    The function with which we handle " standard " input in " C " programming is " scanf ()". For an integer variable, for example, " x " the function structure is as follows:

```
scanf ("%d",& x );
```

    For arguments " scanf ()" takes the data type of the variable (in the example, "% d " where it is an integer variable) and its address with the "&" operator (& x).
" scanf ()" essentially binds to the input channel so that the user can assign the desired data to the variable that the function calls.

d) <u>What do you know about the functions with which we handle " standard " output in " C " programming?</u>

The function with which we handle the " standard " output in " C " programming is " printf ()". We want to print, for example, the message " Hello world » the structure of the function is as follows:

```
print f (" Hello world ");
```

  " printf ()" is connected to the output channel to print the message " Hello world » in it.
    For an integer variable " x ", the function structure is as follows:

```
printf (" x is: %d", x );
```

    For arguments " printf ()" takes the data type of the variable (in the example, "% d " where it is an integer variable) and the variable itself (x).
' printf ()' is attached to the output channel so that it prints with the appropriate message (X is:...), the contents of the variable ' x ' inside it.

COMPUTER PROGRAMMING

<u>**I EMA 3**</u>

a) <u>What are numeric, relational and logical operators?</u>

ARITHMETIC FUNCTIONS

     Arithmetic operators implement a mathematical operation, namely the operation of addition, subtraction, multiplication and division (quotient and remainder) between the contents of two or more variables and produce the corresponding result. Table "3.1" presents in detail for two integer variables " x " and " y ", the operators, their structure, how they are pronounced and their function.

PERFORMER	STRUCTURE	PRONUNCIATION	OPERATION
+	x + y	x plus y	Adds x to y
-	x - y	x minus y	Subtracts x from y
*	x * y	x times y	Multiply x by y
/	x / y	x div y	Calculates the quotient of division by divisor x and divisor y
%	x % y	x mod y	Calculates the remainder of the division by divisor x and divisor y

- Table 3.1 -

OPERATORS

Relational operators compare the contents of two or more variables. In fact, the result they produce is "1" (True), when the condition is true and "0" (False), when the condition is false. For example, the condition "5>3" is true (True), so the value "1" will be produced, while, the condition "10==3" is false (False), so the value "0" will be produced. Table "3.2" presents in detail for two integer variables " x " and " y ", the operators, their structure, how they are pronounced and their operation.

PERFORMER	STRUCTURE	PRONUNCIATION	OPERATION
==	x == y	x equals y	Compares whether x and y are equal
!=	x ! = y	x different from y	Compares if x and y are not equal
<	x < y	x less than y	Compares whether x is less than y
>	x > y	x greater than y	Compares whether x is greater than y
<=	x < = y	x less than or equal to y	Compares whether x is less than or equal to y
>=	x > = y	x greater than or equal to y	Compares whether x is greater than or equal to y

- Table 3.2 -

PERFORMER	STRUCTURE	PRONUNCIATION	OPERATION
AND	x && y	x AND y	Checks x against y to produce the value 1(True) or 0(False) (table 3.4.1)
OR	x || y	x OR y	Checks x against y to produce the value 1(True) or 0(F alse) (table 3.4.2)
NOT	! x	NO x	Tests x to produce the value 1(True) or 0(False)

			(table 3.4.3)

REASONABLE EXECUTIVES

The logical operators check the contents of two or more variables and from the tables "3.4.1", "3.4.2", "3.4.3" (Truth tables) produce the corresponding result "1" (T rue) or "0" (False) . Table "3.3" presents in detail for two integer variables " x " and " y ", the operators, their structure, how they are pronounced and their function.

- Table 3.3 -

AND **truth tables**

x	y	x && y
1	1	1
0	1	0
1	0	0
0	0	0

- Table 3.4.1 -

OR

x	y	x || y
1	1	1
0	1	1
1	0	1
0	0	0

- Table 3.4.2 -

NOT

x	! x
1	0
0	1

- Table 3.4.3 -

b) <u>What is the difference between the preincrement operator and the postincrement operator?</u>

The preincrement operator (++ x) and the postincrement operator (x ++) take the contents of a variable and increment it by one unit (integer value). Their difference is in the result they produce. For example, we have the variable " x = 1", the command "++ x " will increase the value of the variable by one and the value "2" will be entered in " x " and the result it

will produce will be the value " 2". Similarly, the command " x ++ " will also increment this variable's value by one and occupy

say to " x " the value "2", but the result it will produce will be the value "1".

SOURCE CODES / DOCUMENTATION

TOPIC 4

LABEL " MathsIntegers . c »

The "Program " MathsIntegers . c "" (Source Code) and the " Documentation " MathsIntegers . c "" (Question, Structure, Variables, Crossing, Examples) answer the question of Topic "4".

PROGRAM " MathsIntegers.c "

```c
1 #include < stdio.h >

2 # include < math.h >

3

4 int main (int argc , char ** argv )

5

6 {

7

8 system (" chcp 1253");

9

1 0 int A, B; // Declaration variables

11 int sum, diff, prod, int_quo , rem;

12 double real_quo ;

13 int power;

14 doubles root ?

15

16 printf ("========================================== ======\ n \ n ");

1 7 printf ("Math operations with integers\ n \ n "); // Title of the
program

18 printf ("============================================= =========\ n
\ n ");

19 printf ("Enter the 1st integer A : ");

20 scanf ("% d ", & A ); // Input the first integer "A"

21 printf ("Enter the 2nd integer B : ");

22 scanf ("% d ", & B ); // Input the second integer "B"

23 printf ("\n-------------------------------- ---------------\n\n");
```

```
24 p rintf ("The 1st integer : [%20d]\n", A); // Print the first integer "A"

25 printf ("The 2nd integer : [%20d]\n\n", B); // Print the second integer
"B"

26 sum = x + y ; // Calculate the sum of the addition

27 diff = x - y ; // Calculate the difference of the subtraction

28 prod = x * y ; // Calculate the product of the multiplication

39 int _ quo = x / y ; // Calculate the integer quotient of division

30 rem = x % y ; // Calculate the remainder of the division

31 printf ("Sum : [%20 d ]\ n ", sum ); // Print the sum of the addition

3 2 printf ("Diff : [%20d]\n", diff); // Print the difference of the
subtraction

33 printf ("Product : [%20d]\n", prod); // Print the product of
multiplication

3 4 printf ("Integer quotient : [%20d]\n", int_quo); // Print the integer
quotient of division

3 5 printf ("Remainder : [%20d]\n", rem); // Print remainder of division

36 real _ quo = ( double ) A / B ; // Calculate the actual quotient of the
division

37 printf ("Real quotient : [%20.6 lf ]\ n ", real _ quo ); // Print the
actual quotient of division

3 8 power = A * A ; // Calculate the square of " A "

39 root = sqrt (( double ) B ); // Calculate the square root of " B "

40 printf ("Square of A : [%20d]\n", power); // Print the square of "A"

41 printf ("Square root of B : [%20.6lf]\n\n", root); // Print the square
root of "B"

4 2 printf ("------------------------------------------------ --------------
-\n\n");

4 3

44 return 0;

45

46 }
```

DOCUMENTATION " MathsIntegers . c »

REQUIRED

COMPUTER PROGRAMMING

The program " MathsIntegers . c " achieves the following functions:

a) Reads from the " standard " input two integers.

b) Calculates the sum, difference, product, integer quotient and remainder of the division of two numbers.

c) Calculates the actual quotient of two numbers.

d) Calculates the square of the first number and the square root of the second.

STRUCTURE

In order to implement the request, the following libraries were initially used:

a) " stdio . h " : Contains the ready-made functions " scanf ()" and " printf () attached to the input and output channels respectively to read and print contents of the respective variables. Also ' printf ()' was used to print characteristic messages for better understanding of the source code.

b) " math . h » : Contains the ready-made math function " sqrt ()" to calculate the square root of the second variable.

Additionally , the characteristic operators were also used:

a) numeric: +, -, *, /, %

b) assignment: =

c) & operator: For the variable address as the second argument of the " standard " input function, " scanf () .

VARIABLES

A antenna variables (of type " int ")

A (The first integer)

B (The second integer)

sum (The sum of the two numbers)

diff (The difference of the two numbers)

prod (The product of the two numbers)

int _ quo (The integer quotient of the division of the two numbers)

rem (The remainder of the division of the two numbers)

power (The square of the first integer)

Double precision real variables (of type " double ")

real _ quo (The real quotient of the division of the two numbers)

root (The square root of the second corresponding real number)

DIVISION

Declaration of variables (lines 10-14)

The program starts with the declaration of variables in lines "10-14" (detailed see subsection "Variables").

Program title (line 17)

In line "17" with a function " printf ()" the title of the program (Mathematical operations with integers) is printed from the " standard " output, as well as two escape characters of this line break (\
n).

Input of the first integer "A" (lines 19-20)

In line "20" with the function " scanf ()" the first integer is read from the " standard " input. Then, with the " printf ()" function, the corresponding message

is printed from the " standard " output (line 19). **Enter the second integer " B " (lines 21-22)**

In line "22" with the function " scanf ()" the second integer is read from the " standard " input. Then, with the " printf ()" function, the corresponding message is printed from the " standard " output (line 21).

Print the first integer "A" (line 24)

In line "24" the content of the variable " A " (the first integer) is printed from the " standard " output with a function " printf ()" accompanied by the appropriate

message . It is worth noting that the alphanumeric

format is "%20 d ", that is, the result will be printed immediately after "20" blank characters. This is to align the results evenly.

Printing the second integer "B" (line 25)

In line "25" the content of the variable " B " (the second integer) is printed from the " standard " output with a function " printf ()" accompanied by the appropriate message . It is worth noting that the alphanumeric format is "%20 d ", that is, the result will be printed immediately after "20" blank characters. This is to align the results evenly.

Calculation of the sum of the addition (line 26)

In line "26" is calculated with the numerical expression " A + B " (addition), the sum of the two integers and the result is registered in the variable " sum ".

Calculation of the difference of subtraction (line 27) In line "27"

the difference of the two integers is calculated with the arithmetic expression " A - B " (subtraction) and the result is stored in the variable " diff ".

Calculation of the product of multiplication (line 28)

In line "2 8" the arithmetic expression " A * B " (multiplication) calculates the product of the two integers and the result is stored in the variable " prod ".

Calculation of the integer quotient of the division (line 29)

In line " 29 " is calculated with the arithmetic expression " A / B " (division), the integer quotient of the two integers and the result is stored in the variable " int _ quo ".

Calculation of the remainder of the division (line 30)

In line " 30 " is calculated with the arithmetic expression " A % B " (division), the remainder of the two integers and the result is registered in the variable " rem ".

Printing the sum of the addition (line 31)

In line " 31" the content of the " standard " output is printed

variable " sum " (the sum of the addition of the two integers) with a function " printf ()" accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20 d ", that is, the result will be printed immediately after "20" blank characters. This is to align the results evenly.

Printing the difference of the subtraction (line 32)

In the line "3 2" the content of the variable " diff " (the difference of the subtraction of the two integers) is printed from the " standard " output with a function " printf ()" accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20 d ", that is, the result will be printed immediately after "20" blank characters. This is to align the results evenly.

Printing the product of the multiplication (line 33)

In the line "3 3" the content of the variable " prod " (the product of the multiplication of the two integers) is printed from the " standard " output with a function " printf ()" accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20 d ", that is, the result will be printed immediately after "20" blank characters. This is to align the results evenly.

Printing the integer quotient of the division (line 34)

In line "3 4" the content of the variable " int _ quo " (the integer quotient of the division of the two integers) is printed from the " standard " output with a function " printf () » accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20 d ", that is, the result will be printed immediately after "20" blank characters. This is to align the results evenly.

Printing the remainder of the division (line 35)

In the line "3 5" the content of the variable " rem " (the remainder of the division of the two integers) is printed from the " standard " output with a function " printf ()" accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20 d ", that is, the result will be printed immediately after "20" blank characters. This is to align the results evenly.

Calculation of the real quotient of the division (line 36)

In line "3 6" is calculated with the arithmetic expression "(double) A / B " (division), the real double-precision quotient of the two integers

thums and the result is stored in the " real _ quo " variable. Since two integers are read from the " standard " input and the calculation of the real double-precision quotient is requested, the integer variables are converted to double-precision reals (double) only for the numerical representation of line "37".

Print the real quotient of the division (line 37)

In line "37" with the function " printf ()" the content of the variable " real _ quo " (the double-precision real, quotient of the division of two) is printed from the " standard " output of integers) accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20.6 lf ", that is, the result will be printed immediately after "20" blank characters with "6" decimal places, since this is a real double precision. This is to align the results evenly.

Calculation of the square of "A" (line 38)

In line "38" the square of the first integer " A " is calculated with the numerical expression " A * A " (power raised to the square) and the result is stored in the integer variable " power ".

Calculation of the square root of "B" (line 39)

In line "39" the arithmetic expression " sqrt ((double) B)" is calculated, the actual double-precision square root of the second integer " B ", and the result is stored in " root " variable . Since the second integer " B " is read as an integer from the " standard " input and the calculation of the double-precision real square root is requested, the integer variable " B " is converted to a double-precision real (double) only for the numerical expression of line "39".

Print the square of "A" (line 40)

In line "4 0" the content of the variable " power " (the power raised to the square of the first integer " A ") is printed from the " standard " output with a function " printf ()" followed by the appropriate message. It is worth noting that the alphanumeric format is "%20 d ", that is, the result will be printed immediately after "20" blank characters. This is to align the results evenly.

Print the square root of "B" (line 41)

In line "41" the content of the variable " root " (the square root of the second integer "B") is printed from the " standard " output with a function " printf ()" accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20.6 lf ", that is, the result will be printed immediately after "20" blank characters with "6" decimal places, since this is a real double precision. This is to align the results evenly.

<u>**EXAMPLES**</u>

<u>Example 1 (A = 10 / B = 5)</u>

===================================== =============

Mathematical operations with integers

== =

Enter the 1st integer A : 10

Enter the 2nd integer B : 5

-- -------------------------------------

The 1st integer A : [10]

The 2nd integer B : [5]

-- ---------------------------------------

Total : [15]

Difference : [5]

Result : [50]

Whole quotient : [2]

Balance : [0]

Actual quotient : [2.000000]

Square of A : [100]

Square root of B : [2.236068]

--- ---

Example 2 (A = -72 / B = -11)

=== =========

Mathematical operations with integers

=== =

Enter the 1st integer A : -72

Enter the 2nd integer B : -11

--- ---

The 1st integer A : [-72]

The 2nd integer B : [-11]

--- ---

Sum : [-83]

Difference : [-61]

Product : [792]

Whole quotient : [6]

Balance : [-6]

Actual quotient : [6.545455]

Square of A : [5184]

Square root of B : [-1.# IND 00]

Example 3 (A = 999999999999 / B = 100000000000)

======================= =======================

Mathematical operations with integers

=== =

Enter the 1st integer A : 999999999999

Enter the 2nd integer B : 100000000000

-- --------------------------------------

The 1st integer A : [-727379969]

The 2nd integer B : [1215752192]

-- --------------------------------------

Total : [488372223]

Difference : [-1943132161]

Product : [-1375135744]

Integer quotient : [0]

Balance : [-727379969]

Actual quotient : [-0.598296]

Square of A : [-139075583]

Square root of B : [34867.638176]

-- --------------------------------------

COMMENTS

The correctness of the results in the above examples was also checked using a standard calculator and a difference was found in "Example 2 (A = -72 / B = -11)" and in "Example 3 (A = 999999999999 / B = 100000000000)" .

Example 1 (A = 10 / B = 5)

In "Example 1" the first integer "10" (A) and the second integer "5" (B) are read from the " standard " input. These two numbers are correctly printed, as well as the results of the addition, subtraction and division of the two numbers (integer quotient and real and remainder), raising the power to the square of the first integer "A" (10) and square root of the second integer "B" (5). The results :

A : 10

B : 5

A + B : 15

A – B : 5

A * B : 50

A / B : 2 (integer)

A % B : 0

A / B : 2.0 00000 (real)

A^2 : 100

$\sqrt{B}$: 2.236068

Example 2 (A == -72 / B == -11)

In "Example 1" the first integer "-72" (A) and the second integer "-11" are read from the " standard " input (B). These two numbers are correctly printed, as well as the results of addition, subtraction and division of the two numbers (whole quotient and real and remainder) and of raising the power to the square of

prime integer 'A' (-72). The result of the square root of the second integer 'B' (-11) in the program is the value '-1.# IND 00'. Obviously, in mathematics the square root of a negative number is not defined in the set of real numbers, but only in the complex ones. The results :

A : -72

B : -11

A + B : -83

A – B : - 61

A * B : 792

A / B : 6 (integer)

A % B : -6

A / B : 6.545455 (real)

A^2 : 5184

$\sqrt{}$ B : -1.# IND 00

Example 3 (A == 999999999999 / B = 100000000000)

In "Example 3" the first integer "999999999999" (A) and the second integer are read from the " standard " input number "100000000000" (B). These two numbers are not printed properly, as they fall outside the value range of the " int " (integer) type of an " N - bit " processor, so completely different numbers that are in that value range are printed. Clearly, the results of addition, subtraction and division of two numbers (integer quotient and real and remainder), raising the power to the square of the first integer "A" (999999999999) and the square root of the second integer " B" (100000000000) is not the desired for these two numbers either. The results refer to the two numbers printed, where 'A' is the number '-727379969' and 'B' is the number '1215752192', where they are correct except for the 'Product' and 'Square of A' which the results produce they are not in the price range. The results produced by the program:

A: -727379969

B : 1215752192

A + B : 488372223

A – B : -1943132161

A * B : -1943132161

A / B : 0 (integer)

A % B : -727379969

A / B : -0.598296 (real)

A ^{2} : -139075583

$\sqrt{}$ B : 34867.638176

The results of the standard calculator for the two disputed results :

A * B : -8.84313791728642e+17

A ^{2} : 1.478053392352805e+18

TOPIC 5

MARK " CubeSphere . " c » The

CubeSphere "Program " . c ”” (Source Code) and the " CubeSphere Documentation" . c ”” (Question, Structure, Variables, Crossing, Examples, Remarks) answer the question of Topic "5".

PROGRAM « CubeSphere . c »

```
1 # include < stdio . h >
2 # define pi 3.14159
3
4 int main (int argc , char ** argv )
5
6
```

```
7

8 system (" chcp 1253");

9

10 doubles x ; // Declare variables

11 double area_cube , volume_cube ;

12 double r;

13 double area_sphere , volume_sphere ;

14

15 printf ("=============================================== ==============
\n\n");

16 printf ("Cube and Sphere Calculations\n\n"); // Program Title

17 printf ("================================================
==================
\n\n");

18 printf ("Enter the cube edge length in meters : ");

19 scanf("%lf", &x); // Input the edge length of a cube in meters

20 printf ("\n-------------------------------- -------------------------
\n\n");

21 printf ("Cube edge length ' m ' : [%20.6lf]\n\n", x);
// Print the length of the edge of the cube in meters

22 area_cube = 6 * (x * x); // Calculate the area of the cube in square
meters

23 vol_cube = x * x * x; // Calculate the volume of the cube in cubic meters

24 printf ("Area of cube in ' m ^2 ' : [%20.6lf]\n", area_cube); // Print the
area of the cube in square meters

25 printf ("Volume of cube in ' m ^3 ' : [%20.6lf]\n\n", vol_cube); // Print
the volume of the cube in cubic meters

26 r = x; // Assign the same value of the length of the edge of the cube to
the length of the radius of a sphere

27 printf ("Radius length of sphere in ' m ' : [%20.6lf]\n\n", r); // Print
the radius length of the sphere in meters

28 area_sphere = 4 * pi * (r * r); // Calculate the area of the sphere in
square meters

29 vol_sphere = (double) 4 / 3 * pi * (r * r * r); // Calculate the volume
of the sphere in cubic meters

30 printf ("Area of sphere in ' m ^2 : [%20.6lf]\n", area_sphere); // Print
the area of the sphere in square meters

31 printf ("Volume of sphere in ' m ^3 ' : [%20.6lf]\n\n", vol_sphere); //
Print the volume of the sphere in cubic meters

32 printf ("----------------------------------------------- -------------
\n\n");
```

33

```
34 return 0;

35

36 }
```

DOCUMENTATION " Geometry . c »

ZHTOYMENO

The CubeSphere program . c " achieves the following functions:

a) Reads from the " standard " input the edge length of a cube in meters.

b) Calculates the area and volume of the cube

c) Assigns the same value given to the edge length of the cube for the radius length of a sphere.

d) Calculates the area and volume of the sphere

STRUCTURE

In order to implement the request, the libraries were initially used :

a) " stdio . h " : Contains the ready-made functions " scanf ()" and " printf () attached to the input and output channels respectively to read and print contents of the respective variables. Also ' printf ()' was used to print characteristic messages for better understanding of the source code.

b) " define ": In line "2" the identification number " pi " and the constant "3.14159" are defined, where during the compilation of the code " pi " will be replaced by the constant "3.14159".

Additionally , the characteristic operators were also used:

a) numeric : *, /

b) assignment : =

c) & operator : For the variable address as the second argument of the " standard " input function, " scanf ().

VARIABLES

Double precision real variables (of type " double ")

x (the edge length of the cube in meters)

area _ cube (the area of the cube)

vol _ cube (the volume of the cube)

r (the radius length of the sphere in meters)

area _ sphere (the area of the sphere)

vol _ sphere (the volume of the sphere)

DIVISION

Declaration of variables (lines 10-13)

The program starts with the declaration of variables in lines "10-13" (for details see subsection "Variables").

Program title (line 16)

In line "16" with a " printf ()" function, the title of the program (Cube and Sphere Calculations) is printed from the " standard "
output , as well as two newline escape characters (\ n).

Enter the length of the edge of a cube in meters (line 19)

In line "19" with the function " scanf ()" read from the " standard " input the length of the edge of a cube in meters. Then, with the " printf ()" function, the corresponding message

is printed from the " standard " output (line 18). **Print the length of the edge of a cube in meters (line 21)**

In line "21" the content of the variable " x " (the edge length of the cube in meters) is printed from the " standard " output with a function " printf ()" accompanied by with the appropriate message. It is worth noting that the alphanumeric format is "%20.6 lf ", that is, the result will be printed immediately after "20" blank characters with "6" decimal places, since this is a real double precision. This is to align the results evenly.

Calculation of the area of the cube in square meters (line 22)

In line "22" it calculates with the numerical expression "6 * (x * x)", the area of the cube in square meters with " x " for its edge length and the result is stored in the variable " area _ cube ".

Calculation of the volume of the cube in cubic meters (line 23)

In line "23" the volume of the cube in cubic meters with " x " as its edge length is calculated with the numerical expression " x * x * x " and the result is registered in the variable " vol_cube "

. Print the area of the cube in square meters (line 24)

In line "24" the content of the variable " area _ cube " (the area of the cube in square meters) is printed from the " standard " output with a function " printf ()" accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20.6 lf ", that is, the result will be printed immediately after "20" blank characters with "6" decimal places, since this is a real double precision. This is to align the results evenly.

Print the volume of the cube in cubic meters (line 25)

In line "25" the content of the variable "vol_cube" (the volume of the cube in cubic meters) is printed from the " standard " output with a function " printf ()" accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20.6 lf ", that is, the result will be printed immediately after "20" blank characters with "6" decimal places, since this is a real double precision. This is to align the results evenly.

Assigning the same value of the length of the edge of the cube to the length of the radius of a sphere (line 26)

In line "26" the variable " r " is assigned the content of the variable " x ", that is, the length of the radius of a sphere has the same value with the edge length of the cube in meters.

Print the length of the radius of the sphere in meters (line 27)

In line "27" the content of the variable " r " (the length of the radius of the sphere in meters) is printed from the " standard " output with a function " printf ()" accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20.6 lf ", that is, the result will be printed immediately after "20" blank characters with "6" decimal places, since this is a real double precision. This is to align the results evenly.

Calculation of the area of the sphere in square meters (line 28)

In line "28" the area of the sphere in cubic meters is calculated with the numerical expression "4 * pi * (r * r)" with " r " for its radius length and the result is recorded in the variable " area _ sphere ".

Calculation of the volume of the sphere in cubic meters (line 29)

In line "29" the numerical expression "(double) 4 / 3 * pi * (r * r * r)" calculates the volume of the sphere in cubic meters with " r " for its radius length and the result is stored in the " vol _ sphere " variable. It is worth noting that the result of the division "4 / 3" was declared as a double-precision real (double), since the variable " vol _ sphere " is a double-precision real.

Printing the area of the sphere in square meters (line 30)

In line "30" the content of the variable " area _ sphere " (the area of the sphere in square meters) is printed from the " standard " output with a function " printf ()" accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20.6 lf ", that is, the result will be printed immediately after "20" blank characters with "6" decimal places, since this is a real double precision. This is to align the results evenly.

Printing the volume of the sphere in cubic meters (line 31)

In line " 31" the content of the variable " vol _ sphere " (the volume of the cube in cubic meters) is printed from the " standard " output with a function " printf ()" accompanied by the appropriate message. It is worth noting that the alphanumeric format is "%20.6 lf ", that is, the result will be printed immediately after "20" blank characters with "6" decimal places, since this is a real double precision. This is to align the results evenly

EXAMPLES

Example 1 (x = 100)

```
===================================
========================
```

Cube and sphere calculations

===
=========

Enter the cube edge length in 'm' : 100

-- -- ---

Cube edge length in 'm' : [100.000000]

Cube area in 'm^2' : [60000.000000]

Cube volume in 'm^3' : [1000000.000000]

Sphere radius length in 'm' : [100.000000]

Sphere area in 'm^2' : [125663.600000]

Volume of sphere in 'm^3' : [4188786.666667]

-- -- ---

Example 2 (x = -60)

=======================================
====================

Cube and sphere calculations

===
=========

Enter cube edge length in 'm' : -60

COMPUTER PROGRAMMING

Cube edge length in 'm' : [-60.000000]

Cube area in 'm^2' : [21600.000000]

Cube volume in 'm^3' : [-216000.000000]

Sphere radius length in 'm' : [-60.000000]

Sphere area in 'm^2' : [45238.896000]

Volume of sphere in 'm^3' : [-904777.920000]

-- -- --
-

Example 3 (x = 67.61925)

==
===================

Y calculations in cube and sphere

==
=========

Enter cube edge length in ' m ' : 67.61925

-- -- --
-

Cube edge length in ' m ' : [67.619250]

Cube area in ' m ^2' : [27434.177823]

Cube volume in ' m ^3' : [309179.754797]

Sphere radius length in ' m ' : [67.619250]

Area of sphere in ' m ^2' : [57457.959139]

Volume of sphere in ' m ^3' : [1295088.034498]

-- -- ---

OBSERVATIONS

The correctness of the results in the above examples was also checked using a standard calculator and no difference was found, immediately there is an observation in "Example 2 (x = -60)".

Example 1 (x = 100)

In "Example 1" the edge length of a cube in meters " x " (100) is read from the " standard " input . The area of the cube in square meters, its volume in cubic meters, the length of the radius of a sphere in meters, the area of the sphere in square meters, and its volume in cubic meters are correctly printed. The results :

x : 100.000000 m

6 * (x * x) : 60000.000000 m ^2

x * x * x : 1000000.000000 m ^3

r : 100.000000 m

4 * π * (r * r) : 125663.600000 m ^2

4 / 3 * π * (r * r *) : 4188786.666667 m ^3

Example 2 (x = -60) In "Example 2"

the edge length of a cube in meters " x " (-60) is read from the " standard " input . Programmatically, the area of the cube in square meters, its volume in cubic meters, the length of the radius of a sphere in meters, the area of the sphere in square meters, and its volume in cubic meters are correctly printed. The same is not true mathematically, as negative values are not used to calculate length, area or volume of shapes. The results :

x : -60.000000 m

6 * (x * x) : 60000.000000 m ^2

x * x * x : 1000000.000000 m ^3

r : -60.000000 m

4 * π * (r * r) : 45238.896000 m ^2

4 / 3 * π * (r * r *) : -904777.920000 m ^3

Example 3 (x = 67.61925) In "Example 3"

the edge length of a cube in meters " x " (67.61925) is read from the " standard " input. The area of the cube in square meters, its volume in cubic meters, the length of the radius of a sphere in meters, the area of the sphere in square meters, and its volume in cubic meters are correctly printed. The results :

x : 67.619250 m

6 * (x * x) : 27434.177823 m ^2

x * x * x : 309179.754797 m ^3

r : 67.619250 m

4 * π * (r * r) : 57457.959139 m ^2

4 / 3 * π * (r * r *) : 1295088.034498 m ^3

COMPUTER PROGRAMMING

Thank you for your attention.