Ministry of Education and Research of the Republic of Moldova

Technical University of Moldova

Department of Software and Automation Engineering

REPORT

Laboratory work nr.1
at Computer Programming

Elaborated by:
st. gr. FAF-231 A. Rudoi

Verified by:
dr., conf. univ. M. Kulev

Chişinău - 2023

Laboratory work nr.1

Topic: Use of control and loop instructions in C language

Purpose of the laboratory work: Studying techniques and methods of use a condition control instructions and cyclic instructions in the C language for tabulating the function.

Problem condition: To calculate and display on the screen the values of the argument x and the values of the function F, defined by 3 given expressions, for the interval $x1 \leq x \leq x2$ and the step px of incrementing the argument x. The values x1, x2, px and the parameters a, b, c are input data of type real.

Variant 7:
Calculation formulas:

$$F = \begin{cases} \dfrac{a + \ln x}{\sin c} - b^2, & for\ x - 2 > 0\ and\ a = 0 \\ \dfrac{x - \sin(x + 1)}{b}, & for\ x - 2 < 0\ or\ a \neq 0 \\ \dfrac{ax + c}{\cos 2x}, & in\ the\ other\ cases \end{cases}$$

Initial data values:
a = 0
b = 1.5
c = 2.5
x1 = 1.5
x2 = 24
px = 2.5

Laboratory work processing:

Short theory on laboratory work topic:

In the realm of computer programming, control and loop instructions are essential components for directing the flow of a program. These instructions allow us to make decisions, repeat actions, and ultimately govern the logic of our code. The utilization of control and loop instructions can greatly enhance the efficiency and functionality of a C program.

Control structures enable programmers to make choices within their code. They can be broadly categorized into three types:

1. **Conditional Statements:** These structures, often implemented using if, else if, and else statements, allow a program to execute different blocks of code based on certain conditions.
2. **Switch Statements:** Switch statements provide a way to choose from multiple code paths based on the value of an expression. They are particularly useful when dealing with a finite set of options.
3. **The ternary operator:** The conditional operator, denoted by ? and :, provides a concise way to write conditional expressions. It's often used for simple conditional assignments or value selections based on a condition.

Loop instructions are indispensable when you need to repeat a particular block of code multiple times. C provides several loop structures:

1. **For Loop:** The for loop is ideal for situations where you know the exact number of iterations required. It typically consists of an initialization, a condition, and an increment or decrement operation.
2. **While Loop:** A while loop repeatedly executes a block of code as long as a specified condition remains true.
3. **Do-While Loop:** Similar to a while loop, a do-while loop executes its block of code at least once, even if the condition is initially false.

The skillful use of control and loop instructions in the C language empowers programmers to create flexible and powerful applications, allowing them to handle diverse scenarios and manage complex logic efficiently.

Description of data (variables):

a) input data:

a, b, c, x1, x2, px - simple variables of real type (float)

b) output data:

x, F – simple variables of real type (float), calculation results;

n – simple variable of integer type (int), the order number of calculation results;

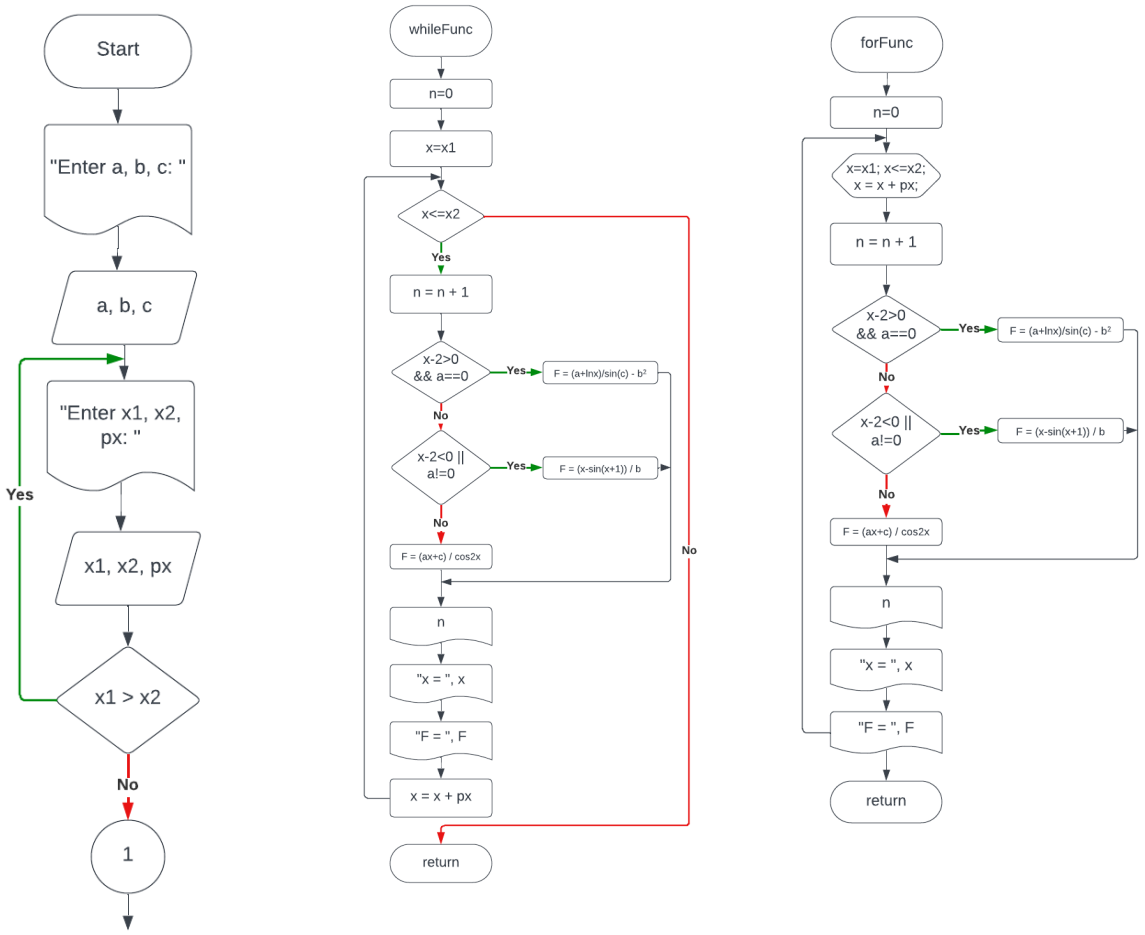
c) working data, being at the same time the output data:

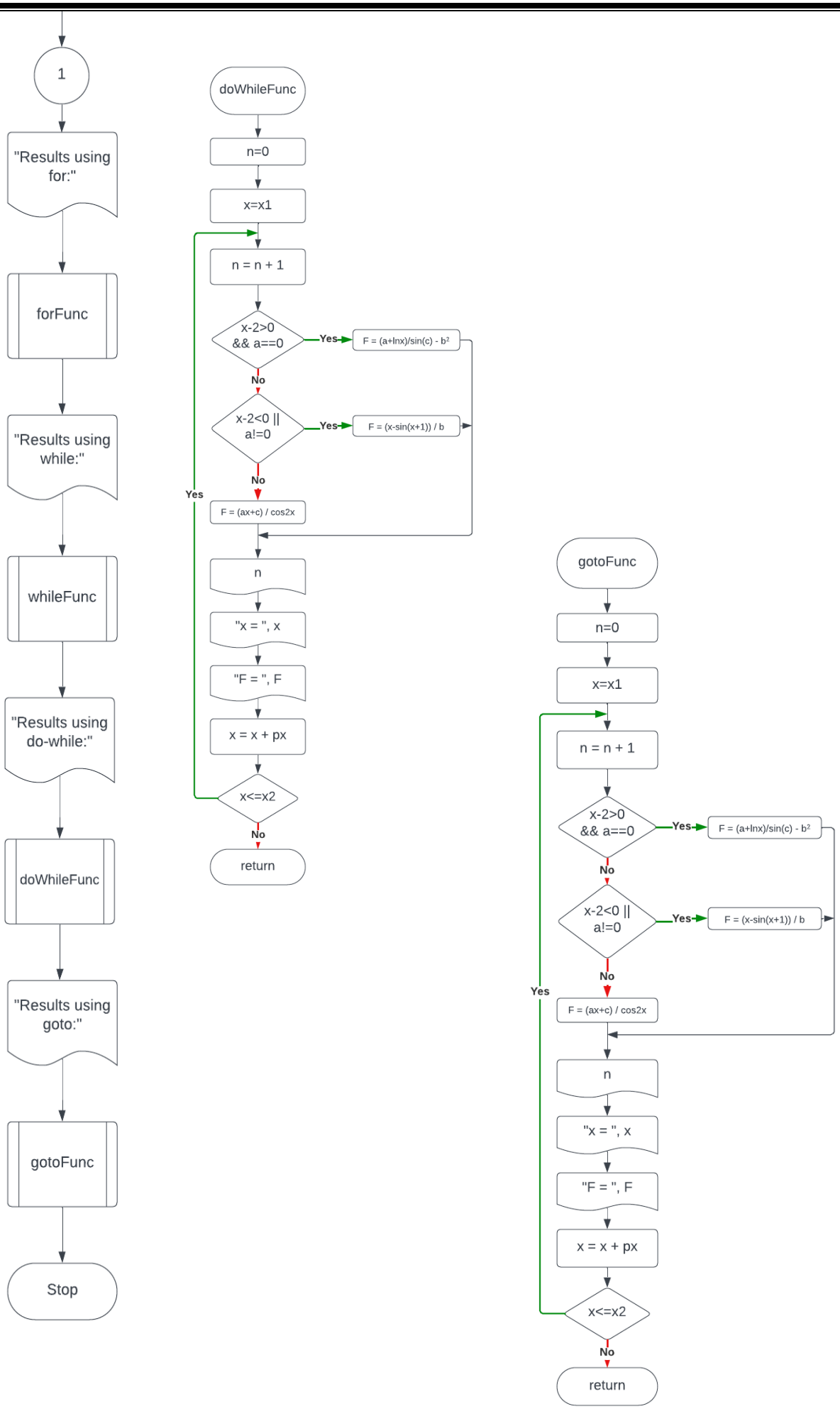
x, F – simple variables of real type (float), results of the calculation;

n – simple variable of integer type (int), the order number of calculation result

Algorithm description:

Flowchart of the algorithm:




1
"Results using for:"
forFunc
"Results using while:"
whileFunc
"Results using do-while:"
doWhileFunc
"Results using goto:"
gotoFunc
Stop
doWhileFunc
n=0
x=x1
n = n + 1
x-2>0 && a==0
Yes
F = (a+lnx)/sin(c) - b²
No
x-2<0 || a!=0
Yes
F = (x-sin(x+1)) / b
No
Yes
F = (ax+c) / cos2x
n
"x = ", x
"F = ", F
x = x + px
x<=x2
No
return
gotoFunc
n=0
x=x1
n = n + 1
x-2>0 && a==0
Yes
F = (a+lnx)/sin(c) - b²
No
x-2<0 || a!=0
Yes
F = (x-sin(x+1)) / b
No
Yes
F = (ax+c) / cos2x
n
"x = ", x
"F = ", F
x = x + px
x<=x2
No
return

Program code in C language (listing of the program):

```
#include <stdio.h>
#include <math.h>

float a, b, c;
float x1, x2, px;
float x, F;
int n;

void forFunc();
void whileFunc();
void doWhileFunc();
void gotoFunc();

int main(){
    printf("Enter a, b, c: ");
    scanf("%f%f%f", &a, &b, &c);
    do {
        printf("Enter x1, x2, px: ");
        scanf("%f%f%f", &x1, &x2, &px);
    } while(x1 > x2);

    printf("\nResults using for: \n");
    forFunc();

    printf("\nResults using while: \n");
    whileFunc();

    printf("\nResults using do-while: \n");
    doWhileFunc();

    printf("\nResults using goto: \n");
    forFunc();

    return 0;
}

void forFunc(){
    n = 0;
    for(x=x1; x<=x2; x+=px) {
        n++;
        if(x - 2 > 0 && a == 0) {
            F = ((a + log(x)) / sin(c)) - pow(b, 2);
        } else if(x - 2 < 0 || a != 0) {
            F = (x - sin(x + 1)) / b;
        } else {
            F = (a*x + c) / cos(2*x);
        }
        printf("%d: x = %f\t F = %f\n", n, x, F);
    }
}
```

```
void whileFunc(){
    n = 0;
    x = x1;
    while(x <= x2) {
        n++;
        if(x - 2 > 0 && a == 0) {
            F = ((a + log(x)) / sin(c)) - pow(b, 2);
        } else if(x - 2 < 0 || a != 0) {
            F = (x - sin(x + 1)) / b;
        } else {
            F = (a*x + c) / cos(2*x);
        }
        printf("%d: x = %f\t F = %f\n", n, x, F);
        x += px;
    }
}

void doWhileFunc(){
    n = 0;
    x = x1;
     do{
        n++;
        if(x - 2 > 0 && a == 0) {
            F = ((a + log(x)) / sin(c)) - pow(b, 2);
        } else if(x - 2 < 0 || a != 0) {
            F = (x - sin(x + 1)) / b;
        } else {
            F = (a*x + c) / cos(2*x);
        }
        printf("%d: x = %f\t F = %f\n", n, x, F);
        x += px;
    } while(x <= x2);
}

void gotoFunc(){
    n = 0;
    x = x1;
    func:
        n++;
        if(x - 2 > 0 && a == 0) {
            F = ((a + log(x)) / sin(c)) - pow(b, 2);
        } else if(x - 2 < 0 || a != 0) {
            F = (x - sin(x + 1)) / b;
        } else {
            F = (a*x + c) / cos(2*x);
        }
        printf("%d: x = %f\t F = %f\n", n, x, F);
        x += px;
   if(x <= x2) {
    goto func;
   }
}
```

Results of running and testing the program:

```
Admin@DESKTOP-3BO5FF3 MINGW64 ~/Desktop/University/PC/Lab-1
$ ./a.exe
Enter a, b, c: 0 1.5 2.5
Enter x1, x2, px: 1.5 24 2.5

Results using for:
1: x = 1.500000  F = 0.601019
2: x = 4.000000  F = 0.066389
3: x = 6.500000  F = 0.877635
4: x = 9.000000  F = 1.421390
5: x = 11.500000         F = 1.830970
6: x = 14.000000         F = 2.159658
7: x = 16.500000         F = 2.434196
8: x = 19.000000         F = 2.669927
9: x = 21.500000         F = 2.876476
10: x = 24.000000        F = 3.060279

Results using while:
1: x = 1.500000  F = 0.601019
2: x = 4.000000  F = 0.066389
3: x = 6.500000  F = 0.877635
4: x = 9.000000  F = 1.421390
5: x = 11.500000         F = 1.830970
6: x = 14.000000         F = 2.159658
7: x = 16.500000         F = 2.434196
8: x = 19.000000         F = 2.669927
9: x = 21.500000         F = 2.876476
10: x = 24.000000        F = 3.060279

Results using do-while:
1: x = 1.500000  F = 0.601019
2: x = 4.000000  F = 0.066389
3: x = 6.500000  F = 0.877635
4: x = 9.000000  F = 1.421390
5: x = 11.500000         F = 1.830970
6: x = 14.000000         F = 2.159658
7: x = 16.500000         F = 2.434196
8: x = 19.000000         F = 2.669927
9: x = 21.500000         F = 2.876476
10: x = 24.000000        F = 3.060279

Results using goto:
1: x = 1.500000  F = 0.601019
2: x = 4.000000  F = 0.066389
3: x = 6.500000  F = 0.877635
4: x = 9.000000  F = 1.421390
5: x = 11.500000         F = 1.830970
6: x = 14.000000         F = 2.159658
7: x = 16.500000         F = 2.434196
8: x = 19.000000         F = 2.669927
9: x = 21.500000         F = 2.876476
10: x = 24.000000        F = 3.060279
```

Verification of the results:

Input	Input	Input	Input	Input
$\frac{1.5 - \sin(1.5 + 1)}{1.5}$	$\frac{0 + \log(4)}{\sin(2.5)} - 1.5^2$	$\frac{0 + \log(6.5)}{\sin(2.5)} - 1.5^2$	$\frac{0 + \log(9)}{\sin(2.5)} - 1.5^2$	$\frac{0 + \log(11.5)}{\sin(2.5)} - 1.5^2$
Result	Result	Result	Result	Result
0.601019…	0.0663891…	0.877635…	1.42139…	1.83097…

Analysis of results and conclusions:

1. This program demonstrates the ability to write and execute C programs effectively, including the use of control and loop instructions.
2. The program's results confirm its correct operation, indicating that it accurately calculates and displays the values of x and the mathematical expression F for the given inputs.
3. The program showcases the use of different loop instructions, namely 'for', 'while', 'do-while' and 'goto' to iterate and calculate F for a range of x values.
4. The advantage of the developed program lies in its implementation of a linear algorithm for calculating F based on various conditions. This simplicity contributes to code readability and maintenance.
5. One notable limitation is that the program does not validate the input data for x1, x2, and px when using the 'do-while' loop. This could potentially lead to unexpected behavior if invalid values are entered.
6. The program can be enhanced by adding input validation operations to ensure that x1 is less than or equal to x2 and that px is a valid step size. This would make the program more robust and user-friendly.

.

Bibliography:

1. Carcea L., Vlas S., Bobicev V. Informatics: Tasks for laboratory works. Chisinau: UTM, 2005. -19 p.
2. The overview of Computer Programming course lessons for students (lecturer: associateprofessor M. Kulev). Chisinau, UTM, FCIM, 2023.
3. https://www.tutorialspoint.com/cprogramming/index.htm
4. https://kremlin.cc/k&r.pdf
5. https://www.wolframalpha.com/