

ECE 220 Computer Systems & Programming

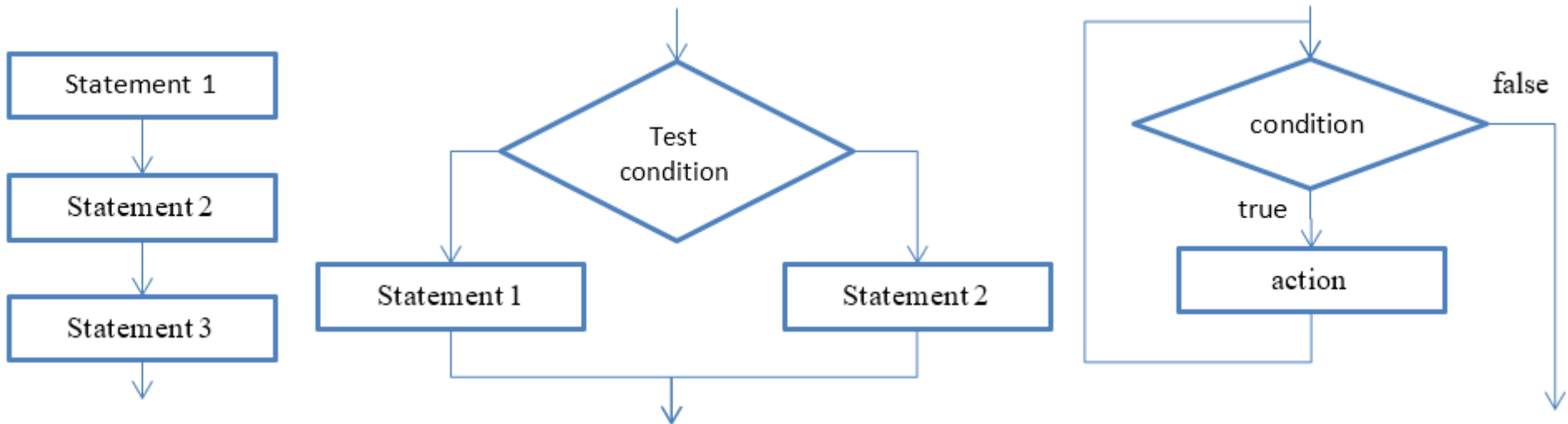
Lecture 8 – Control Structures

February 7, 2019



Control Structure

- There are three basic programming constructs: sequential, conditional, iterative
- Sequential construct means that C program instructions (statements) are executed sequentially, one after another
- Conditional construct means that one or another statement will be executed, but not both, depending on some condition.
- Iterative construct means that some statements will be executed multiple times until some condition is met



Control Structures

Conditional Constructs

- if
- if - else
- switch

Iteration Constructs (loops)

- while
- do - while
- for

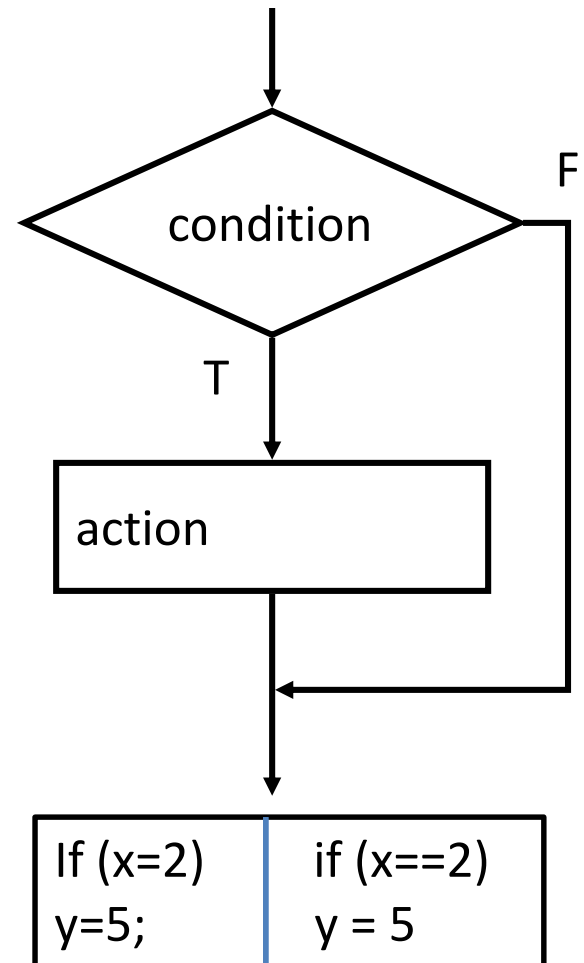
The if Statement (similar to BR in LC-3)

```
int x;  
... //assign some value to x  
if (x < 0)  
    x = -x; //invert x only if x < 0
```

```
int y = 0;  
if ((x > 5) && (x < 25))  
{  
    y = x * x + 5;  
    printf("y = %d\n", y);  
}
```

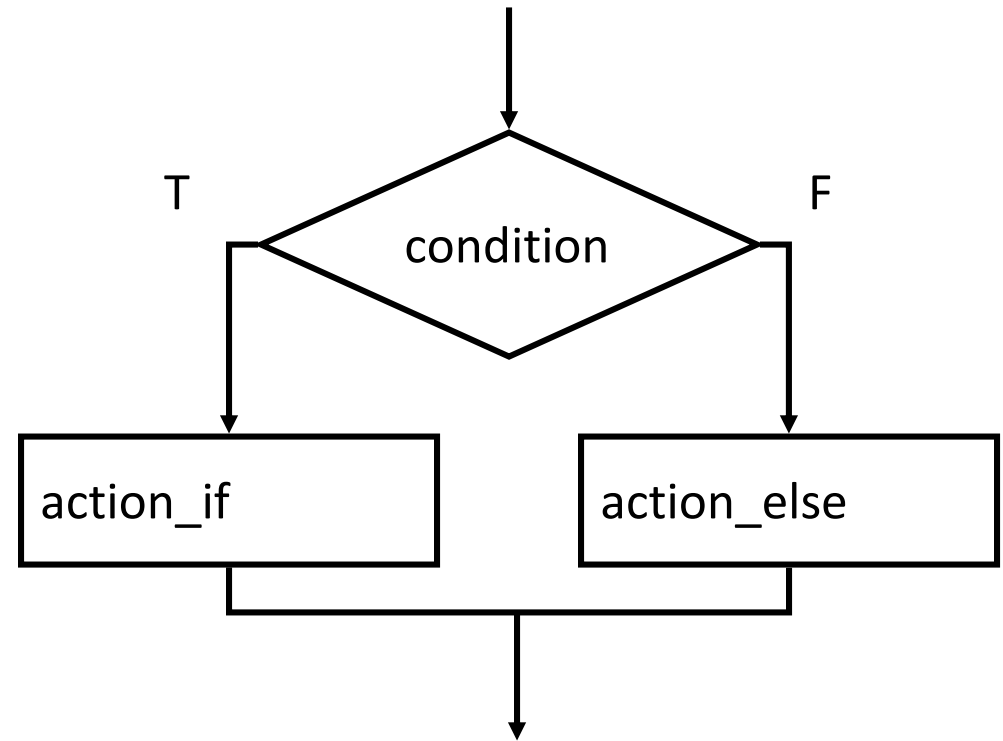
* What would happen if {} is omitted?

```
; LC-3 assembly  
;  
; generate condition code  
;  
BR(nzp) FALSE  
;  
; action  
;  
FALSE  
;
```



The if - else Statement

```
/*x and y are of type int*/  
if (x < 0)  
    x = -x;  
else  
    x = x * 2;  
  
if ((x > 5) && (x < 25))  
{  
    y = x * x + 5;  
    printf("y = %d\n", y);  
}  
else  
    printf("x = %d\n", x);
```



```
; LC-3 assembly  
;  
; generate condition code  
;  
BR(nzp) FALSE  
;  
; action 1  
BRnzp DONE  
;  
FALSE  
; action 2  
;  
DONE
```

If, else-if, else statements:

```
#include <stdio.h>

int main()
{
    int month;

    printf("Enter the number of the month: ");
    scanf("%d", &month);

    if (month == 4 || month == 6 || month == 9 || month == 11)
        printf("The month has 30 days\n");
    else if (month == 1 || month == 3 || month == 5 ||
             month == 7 || month == 8 || month == 10 || month == 12)
        printf("The month has 31 days\n");
    else if (month == 2)
        printf("The month has either 28 days or 29 days\n");
    else
        printf("Don't know that month\n");
}
```

Switch statement:

Using cascaded **if-else** statements

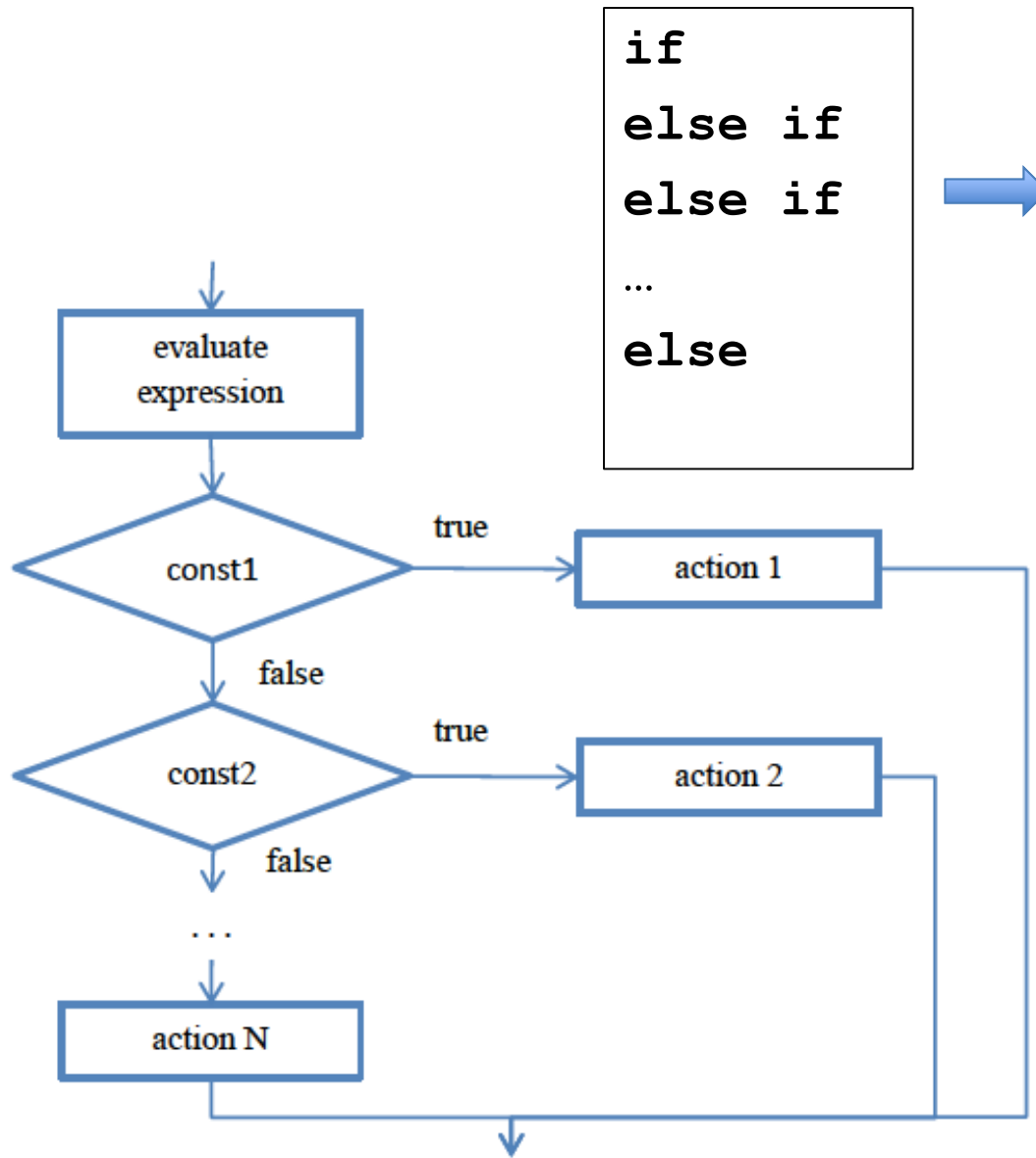
```
if (expression == const1)
    action1;
else if (expression == const2)
    action2;
else if (expression == const3)
    action3;
...
else
    actionN;
```

Using **switch** statement

```
switch (expression) {
    case const1:
        action1;
        break;
    case const2:
        action2;
        break;
    case const3:
        action3;
        break;
    ...
    default:
        actionN;
}
```

- gives compiler an opportunity to better optimize the code by bypassing some testing.
- e.g. expression is a keypress data [see the example code (switch.c) on github]

The switch Statement



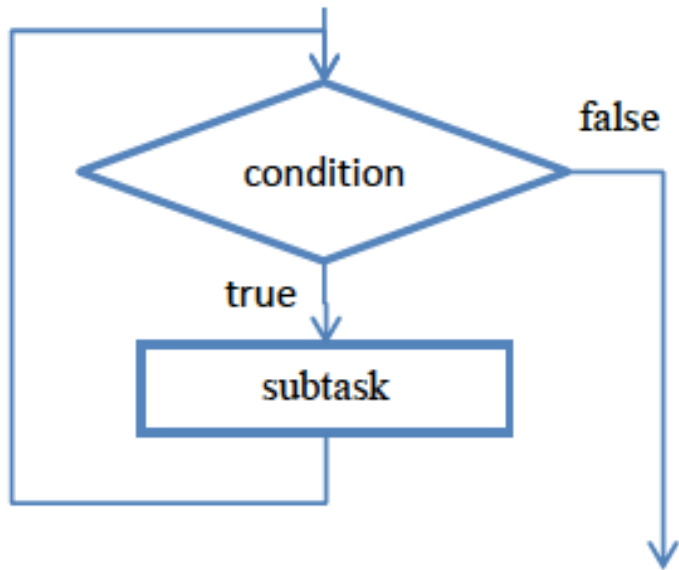
```
if  
else if  
else if  
...  
else
```

```
switch (expression)  
{  
    case const 1:  
        action 1;  
        break;  
    case const 2:  
        action 2;  
        break;  
    ...  
    default:  
        default action;  
        break;  
}  
  
// notice the use of 'break'
```

***See the github example code**

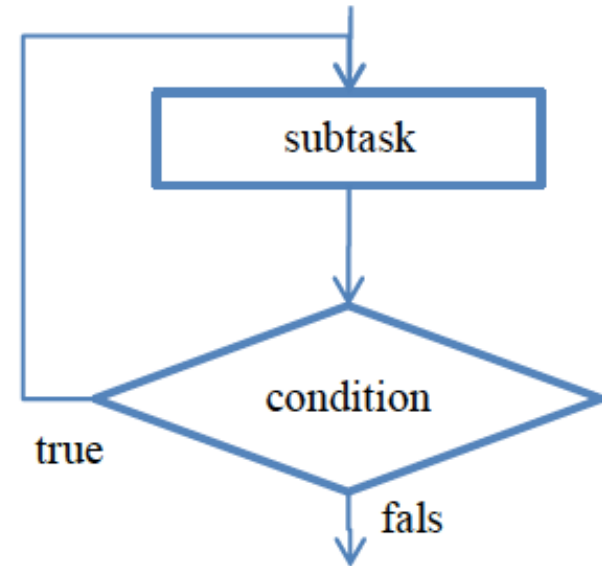
The while / do - while Statement

while: loop body may or may not be executed even once



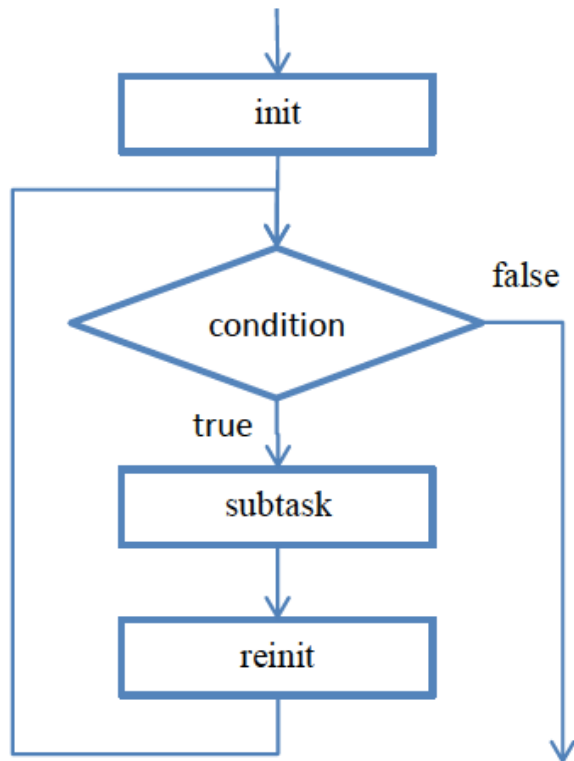
```
int x = 0;
while (x < 10) {
    printf("x=%d\n", x);
    x = x + 1;
}
```

do – while: loop body will be executed at least once



```
int x = 0;
do {
    printf("x=%d\n", x);
    x = x + 1;
} while (x < 10);
```

The for Statement



```
int x = 0;
while (x < 10) {
    printf("x=%d\n", x);
    x = x + 1;
}
```

```
int x;
for (x = 0; x < 10; x++)
{
    printf("x=%d\n", x);
}
```

➤ What would cause while loop or for loop to become infinite loops?

```
for (x = 0; x < 10; x++) {
    if (x == 5)
        break;
    printf("x=%d\n", x);
} /* what would be the print out? What if
'break' is replaced with 'continue'? */
```

Example: on github
break_continue.c

Nested Loops

inner loop is nested within the outer loop **for** ()

```
#include <stdio.h>

int main()
{
    int sum = 0;           /* Initial the result variable */
    int input;             /* Holds user input */
    int inner;             /* Iteration variables */
    int outer;

    /* Get input */
    printf("Input an integer: ");
    scanf("%d", &input);

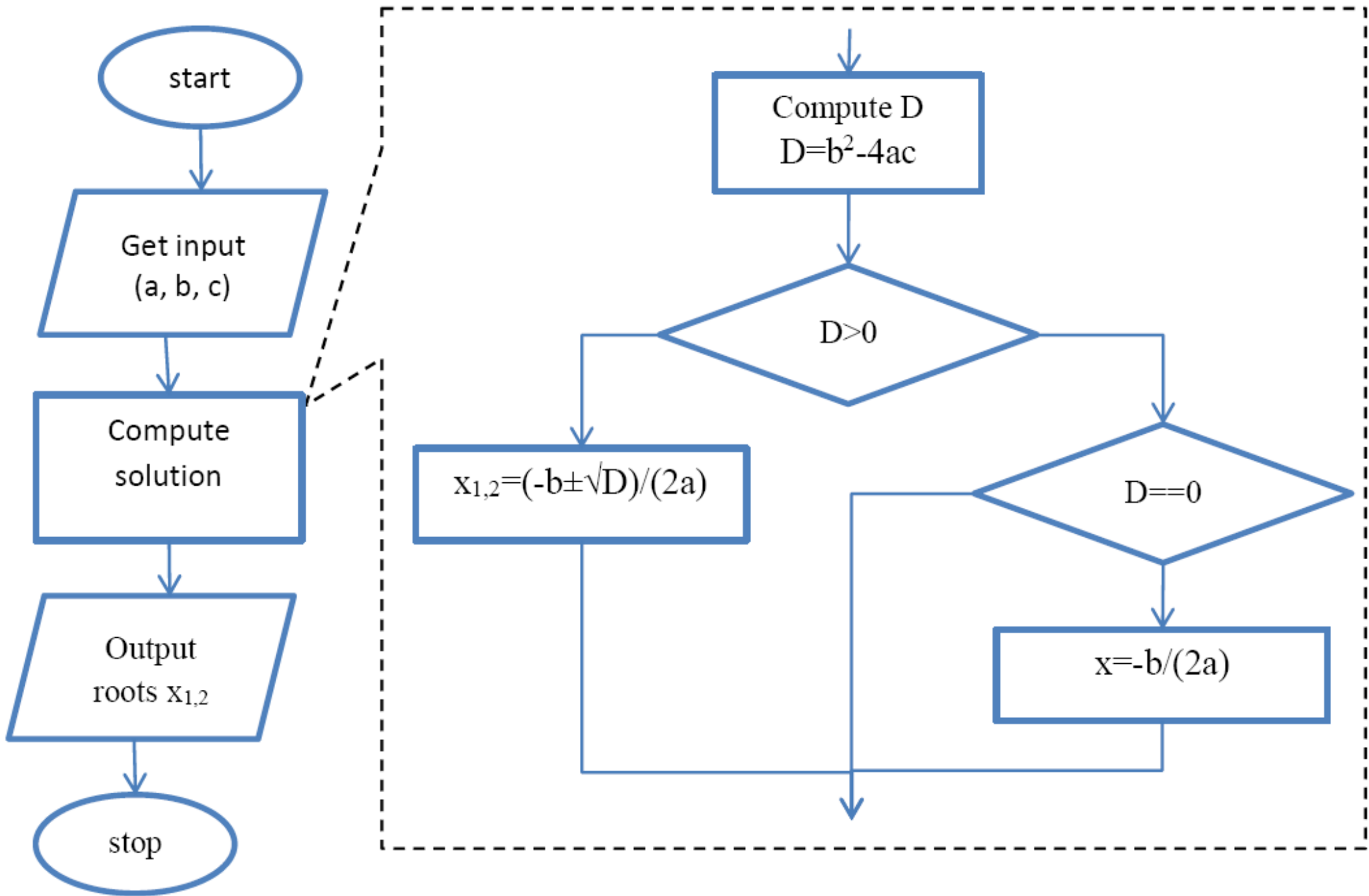
    /* Perform calculation */
    for (outer = 1; outer <= input; outer++)
        for (inner = 0; inner < outer; inner++) {
            sum += inner;
        }

    /* Output result */
    printf("The result is %d\n", sum);
}
```

Example: Computing solution of a quadratic equation $ax^2+bx+c=0$

- **Algorithm:**
 - $D = b^2 - 4ac$
 - If D equals 0, there is one real root: $x = -b/(2a)$
 - If D is positive, there are two roots: $x_{1,2} = (-b \pm \sqrt{D})/(2a)$
 - If D is negative, no real roots exist
- Problem decomposition into separate steps using a flowchart
 - Get input
 - Compute solution according to the above algorithm
 - Print output

Adapted from V. Kindratenko's notes



Solution of the quadratic equation:

Adapted from V. Kindratenko's notes

```
/* solution of the quadratic equation  $ax^2+bx+c=0$  */

#include <stdio.h>          /* needed for printf and scanf */
#include <math.h>           /* needed for sqrtf */

int main()
{
    float a, b, c;         /* quadratic equation coefficients */
    float D;               /* determinant */
    float x1, x2;          /* solution(s) */

    /* get equation coefficients */
    printf("Enter a, b, and c: ");
    scanf("%f %f %f", &a, &b, &c);
    printf("Solving equation  $%fx^2+%fx+%f=0$ \n", a, b, c);

    /* compute solution */
    D = b * b - 4 * a * c;  /* compute determinant */

    if (D > 0)              /* two real roots exist */
    {
        x1 = (-b + sqrtf(D)) / (2 * a);
        x2 = (-b - sqrtf(D)) / (2 * a);
    }
}
```

(continue)

```
}
else if (D == 0)                /* only one root exists */
    x1 = -b / (2 * a);

/* print results */
if (D > 0)
    printf("x1=%f, x2=%f\n", x1, x2);
else if (D == 0)
    printf("x=%f\n", x1);
else
    printf("No real roots exist\n");

return 0;
}
```

- To compile, we will need to link the code with additional library (libm.a) using **-lm** compiler flag
 - `gcc -Wall -ansi -pedantic -lm -o quadratic quadratic_equation.c`
- Examples:
 - $x^2+2x-8=0$: $x_1 = -0.5$, $x_2 = -1.5$
 - $x^2-10x+25=0$: $x = 5$
 - $5x^2-2x+2=0$: no real roots

Exercise

Write a program to print an $n \times n$ identity matrix using nested loops.
(Adapted from Yuting's notes)

```
#include <stdio.h>
#define N 5
int main(){

}
}
```


Follow-up Questions

- What are some ways to stop after printing the second '1' on the main diagonal, such as the example below?
1 0 0
0 1
- How to take user input for the value of n , for which n has to be >0 and <10 ?
(If user input is invalid, print the message “Number entered is invalid” and prompt the user to enter a number again.)