Withdraw example.

```c
#include "stdio.h"
#include "conio.h"
int main() {
    int a;  // amount of money
    int p100, p50, p20, p10, p5;
    printf("How much do you withdraw? ");
    scanf("%d", &a);
    p100 = a/100; a = a%100;
    p50 = a/50; a = a%50;
    p20 = a/20; a = a%20;
    p10 = a/10; a = a%10;
    p5 = a/5; a = a%5;  // none paid
    printf("\n%d\n", p100, p50, p20, p10, p5);
    printf("%d can not paid", a);
    getch();
    return 0;
}
```

**if Selection Statement**

Programs use selection statements to choose among alternative courses of action. For example, suppose the passing grade on an exam is 60. The pseudocode statement

*If student's grade is greater than or equal to 60 Print "Passed"*

determines whether the condition "student's grade is greater than or equal to 60" is true or false.
If student’s grade is greater than or equal to 60
Print "Passed"

If the condition is true, then "Passed" is printed and the next pseudocode statement in order is "performed" (pseudocode is not a real programming language). If the condition is false, the print statement is ignored and the next pseudocode statement in order is performed.
The preceding pseudocode If statement can be written in C++ as

```c++
if (grade >= 60)
    printf("Passed");
```

**if...else Double-Selection Statement**

The if single-selection statement performs an indicated action only when the condition is **True**; otherwise the action is skipped. The **if...else** double-selection statement allows the programmer to specify an action to perform when the condition is **true** and a **different action** to perform when the condition is **false**.
For example, the pseudocode statement

If student's grade is greater than or equal to 60
  Print "Passed"
Else
  Print "Failed"

prints "Passed" if the student's grade is greater than or equal to 60, but prints "Failed" if the student's grade is less than 60. In either case, after printing occurs, the next pseudocode statement in sequence is "performed."
The preceding pseudocode If...Else statement can be written in C++ as

```c++
if (grade>=60)
    printf("Passed");
else
    printf("Failed");
```

Let's write a program to calculate final grade of a student and decide “passed” or “failed. 2 midterms (%40) and final (%60)

```c
#include"stdio.h"
#include"conio.h"
int main()
{
    int v1,v2,final;
    float ort;
    printf("enter midterms and final(seperate with , ) ");
    scanf("%d,%d,%d", &v1,&v2,&final);
    ort=(v1+v2)/2*.40+final*.60;
    printf("grade is %f so you are ",ort);
    if (ort>=70)
        printf("Passed");
    else
        printf("Failed");
    getch();
}  
```
Enter midterms and final (seperate with ,) 21,30,60
Grade is 46.000000 so you are Failed

If you input midterms 21,30 and final 60, your output will be above. But in real
(21+30)/2=25.5
25.5x0.4=10.2
10.2+60x0.60=46.2 ???

In C++ summation of two int will be int, (21+30)
Dividing of two int will be int. so 51/2 will not be 25.5 will be 25. To handle this problem, line 8 must be changed with following

```
8     ort=(v1+v2)/2.*.40+final*.60;
```

---

**Conditional Operator (?:)**

C++ provides the conditional operator (?:), which is closely related to the if...else statement. The conditional operator is C++'s only ternary operator it takes three operands.

The operands, together with the conditional operator, form a conditional expression. The first operand is a condition, the second operand is the value for the entire conditional expression if the condition is True and the third operand is the value for the entire conditional expression if the condition is false.
For example, the output statement

```c
printf(grade>=60 ? "Passed" : "Failed");
```

It is equivalent to

```c
if (grade>=60)
    printf("Passed");
else
    printf("Failed");
```

Let's re-write previous example with conditional operator

```c
#include"stdio.h"
#include"conio.h"
int main()
{
    int v1,v2,final;
    float ort;
    printf("enter midterms and final(separate with , ) ");
    scanf("%d,%d,%d", &v1, &v2, &final);
    ort=(v1+v2)/2.*.40+final*.60;
    printf("grade is %f so you are ", ort);
    printf(ort>=70 ? "Passed" : "Failed");
    getch();
}
```
Nested if...else Statements

Nested if...else statements test for multiple cases by placing if...else selection statements inside other if...else selection statements. For example, the following pseudocode if...else statement prints A for exam grades greater than or equal to 90, B for grades in the range 80 to 89, C for grades in the range 70 to 79, D for grades in the range 60 to 69 and F for all other grades:

If student's grade is greater than or equal to 90
Print "A"
Else
   If student's grade is greater than or equal to 80
      Print "B"
   Else
      If student's grade is greater than or equal to 70
         Print "C"
      Else
         If student's grade is greater than or equal to 60
            Print "D"
         Else
            Print "F"
This pseudocode can be written in C++ as

```c
if ( Grade >= 90 ) // 90 and above gets "A"
   printf("A");
else if ( Grade >= 80 ) // 80-89 gets "B"
   printf("B");
else if ( Grade >= 70 ) // 70-79 gets "C"
   printf("C");
else if ( Grade >= 60 ) // 60-69 gets "D"
   printf("D");
else // less than 60 gets "F"
   printf("F");
```

A nested if...else statement can perform much faster than a series of single-selection if statements because of the possibility of early exit after one of the conditions is satisfied.
Dangling-else Problem

The C++ compiler always associates an else with the immediately preceding if unless told to do otherwise by the placement of braces ({ and }). This behavior can lead to what is referred to as the dangling-else problem. For example,

```c
if ( x > 5 )
    if ( y > 5 )
        printf("x and y are > 5");
else
    printf("x is <= 5");
```

appears to indicate that if \(x\) is greater than 5, the nested if statement determines whether \(y\) is also greater than 5. If so, "x and y are > 5" is output. Otherwise, it appears that if \(x\) is not greater than 5, the else part of the if...else outputs "x is <= 5".
Beware! This nested if...else statement does not execute as it appears. The compiler actually interprets the statement as

```c
if ( x > 5 )
    if ( y > 5 )
        printf("x and y are > 5");
    else
        printf("x is <= 5");
```

To force the nested if...else statement to execute as it was originally intended, we must write it as follows:

```c
if ( x > 5 ) {
    if ( y > 5 )
        printf("x and y are > 5");
} else
    printf("x is <= 5");
```

The braces ({}) indicate to the compiler that the second if statement is in the body of the first if and that the else is associated with the first if.
Blocks

The if selection statement normally expects only one statement in its body. Similarly, the if and else parts of an if...else statement each expect only one body statement. To include several statements in the body of an if or in either part of an if...else, enclose the statements in braces ({{ and }}). A set of statements contained within a pair of braces is called a compound statement or a block. We use the term "block" from this point forward.

The following example includes a block in the else part of an if...else statement.

```c
if ( studentGrade >= 60 )
    printf("Passed.\n");
else
{
    printf("Failed.\n");
    printf("You must take this course again.\n");
}
```
In this case, if `studentGrade` is less than 60, the program executes both statements in the body of the `else` and prints

**Failed.**

**You must take this course again.**

Notice the braces surrounding the two statements in the `else` clause. These braces are important. Without the braces, the statement

```c
printf("You must take this course again.\n");
```

would be outside the body of the `else` part of the `if` and would execute regardless of whether the grade is less than 60. This is an example of a logic error.

---

Placing a semicolon ( ; ) after the condition in an `if` statement leads to a logic error in single-selection `if` statements and a syntax error in double-selection `if...else` statements (when the `if` part contains an actual body statement).
Lets write program of finding roots of 2nd degree polynomial

S.1. get coefficients \(a, b, c\)

S.2. calculate \(\Delta\) \((\Delta = b^2 - 4ac)\)

S.3. if \(\Delta \geq 0\) then

S.3.1 calculate \(x_1 = \frac{-b}{2a} + \frac{\sqrt{\Delta}}{2a}\)

S.3.2 calculate \(x_2 = \frac{-b}{2a} - \frac{\sqrt{\Delta}}{2a}\)

S.3.3 show \(x_1\) and \(x_2\)

Else

S.3.1 calculate real part \(r = \frac{-b}{2a}\)

S.3.2 calculate imaginary part \(s = \frac{\sqrt{-\Delta}}{2a}\)

S.3.3 show \(r + is\) and \(r - is\)

---

```c
#include "stdio.h"
#include "conio.h"
#include "math.h"

int main()
{
    float a, b, c, delta, x1, x2, r, im;
    printf("enter the coefficients ");
    scanf("%f,%f,%f", &a, &b, &c);
    delta=b*b-4*a*c;
    if (delta>=0) {
        x1=(-1*b+sqrt(delta))/(2*a);
        x2=(-1*b-sqrt(delta))/(2*a);
        printf("x1=%f\tx2=%f", x1, x2);
    }
    else {
        r=-1*b/(2*a);
        im=sqrt(-1*delta)/(2*a);
        printf("x1=%f+i %fn", r, im);
        printf("x2=%f-i %fn", r, im);
    }
    getch();
} 
```
After calculation of \textit{delta} in line 8, if it is
greater or equal to 0 lines 10,11,12 will be
executed, if not lines 15,16,17,18 will be
executed.

As you see, we add another header file
“math.h” in line 3. This header contains
mathematical functions such that \textit{sin}(x),
\textit{cos}(x), \textit{exp}(x) \left( e^x \right), \textit{sqrt}(x) \left( \sqrt{x} \right), \textit{log}(x)
(natural logarithm), \textit{pow}(a,b) \left( a^b \right)