C Programming

04: Conditional Statements & selection structure

Recap

- Last week
  - Variables
  - Expressions--statement
- This week
  - Branching & looping statements

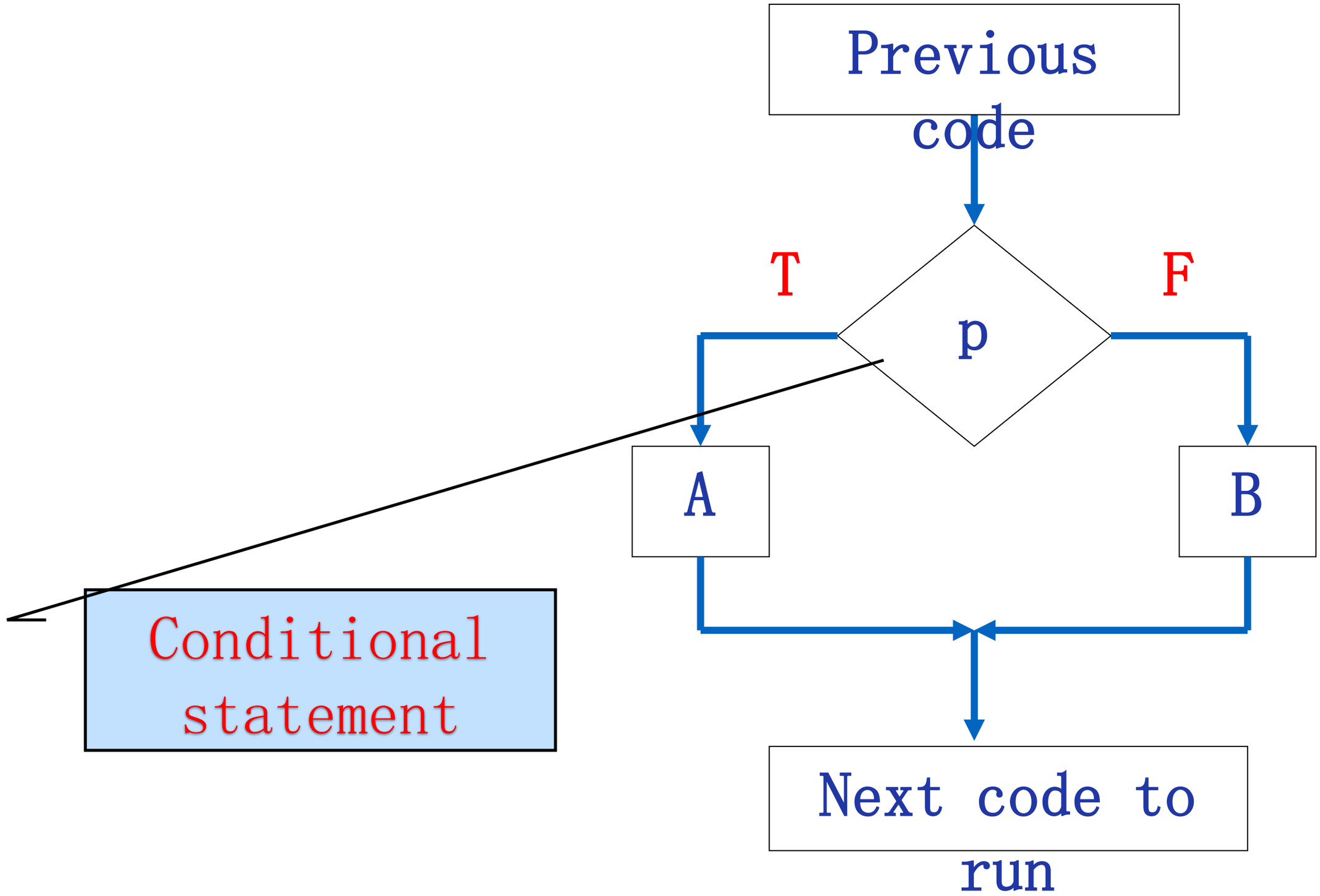
Grade Values

- A professor generates letter grades using the following table.
- Goal: given the scores, to print the grade for each student.

Score	Grade
0-60	F
61-70	D
71-80	C
81-90	B
91-100	A

```
int main()
{       ...
        printf("F\n");
        printf("D\n");
        printf("C\n");
        printf("B\n");
        printf("A\n");
}
```

selection structure



Conditional Statements

- Conditional statements allow us to *decide*
  - Do we take the bus to class?
  - Or do we walk?
- And to execute different statements
  - Put bus pass in pocket
  - Put walking shoes on

Conditions

- A condition is either false or true
  - In C, an integer
    - False = 0
    - True = anything else
- So any expression can be used
  - if it returns an integer or boolean value

Boolean Value

- C programming language assumes any non-zero and non-null values as true, and if it is either zero or null, then it is assumed as false value

Conditional Operators

- We have to be true OR false
- Relational Operators

	True (1)	False (0)
A ==B	A, B are equal	A, B are not equal
A != B	A, B are not equal	A, B are equal
A < B	A is less than B	A is > or = B
A <= B	A is < or = B	A is greater than B
A > B	A is greater than B	A is < or = B
A >= B	A is > or = B	A is less than B

Logical Operators

- Once you have logical values
  - You can start combining them
  - There are three notations for it:

    && (and), || (or), ! (not)

Logical &&, ||, !

- If you have more than one, use ()

A && B	**True if both A and B are true**
A || B	**True if either A or B is true (or both)**
! A	**Unary not - true if A is false**

```
int ReadToGo=
  ((month == may) && (year == 2018)
  && (havePassedEverything));
```

Conditional Expressions

`expr1 == expr2`	tests if `expr1` is equal to `expr2`
`expr1 != expr2`	tests if `expr1` is not equal to `expr2`
`expr1 < expr2`	tests if `expr1` is less than `expr2`
`expr1 <= expr2`	tests if `expr1` is less than or equal to `expr2`
`expr1 > expr2`	tests if `expr1` is greater than `expr2`
`expr1 >= expr2`	tests if `expr1` is greater than or equal to `expr2`
`!expr`	computes the logical NOT of `expr`
`expr1 && expr2`	computes the logical AND of `expr1` and `expr2`
`expr1 || expr2`	computes the logical OR of `expr1` and `expr2`

Operator Priority

Relational Operators	
>	<
>=	<=
==	!=

Logical Operators		
&&	||	!

!	high
Arithmetic operator	
Relation operator	
&& ||	
Assignment operator	low

Selection structure

1

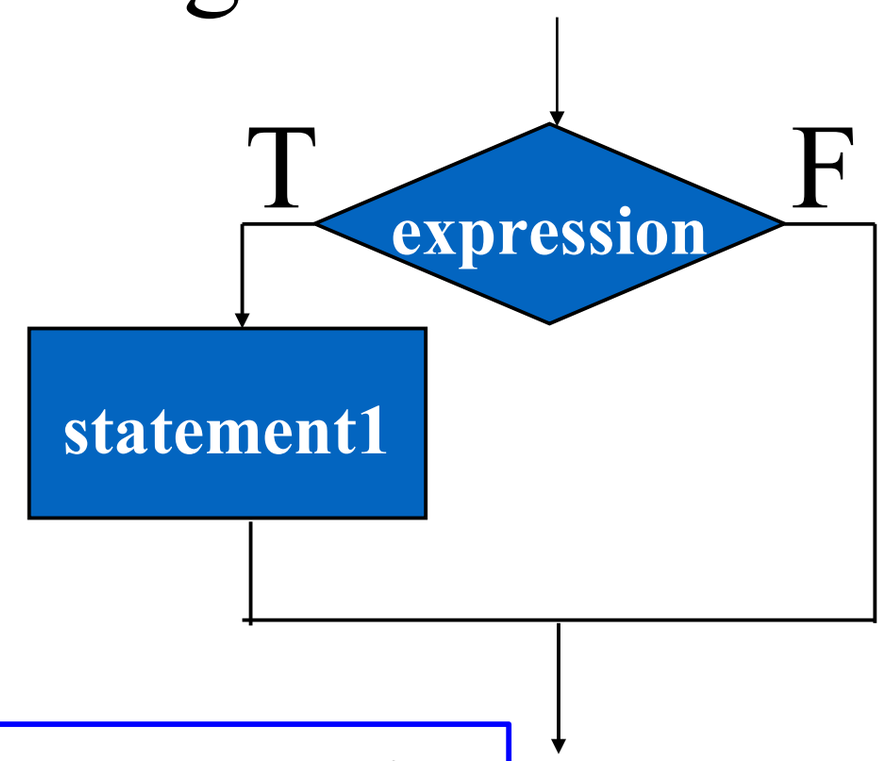
- **Single-alternative selection structure**

2

- **Dual-alternative selection structure**

Single-alternative selection structure

- The single-alternative selection structure evaluates a condition and executes a block of code if that condition is true. If the condition is false, then the block of code is ignored.



An example of a single-alternative selection structure is the ***if*** statement.

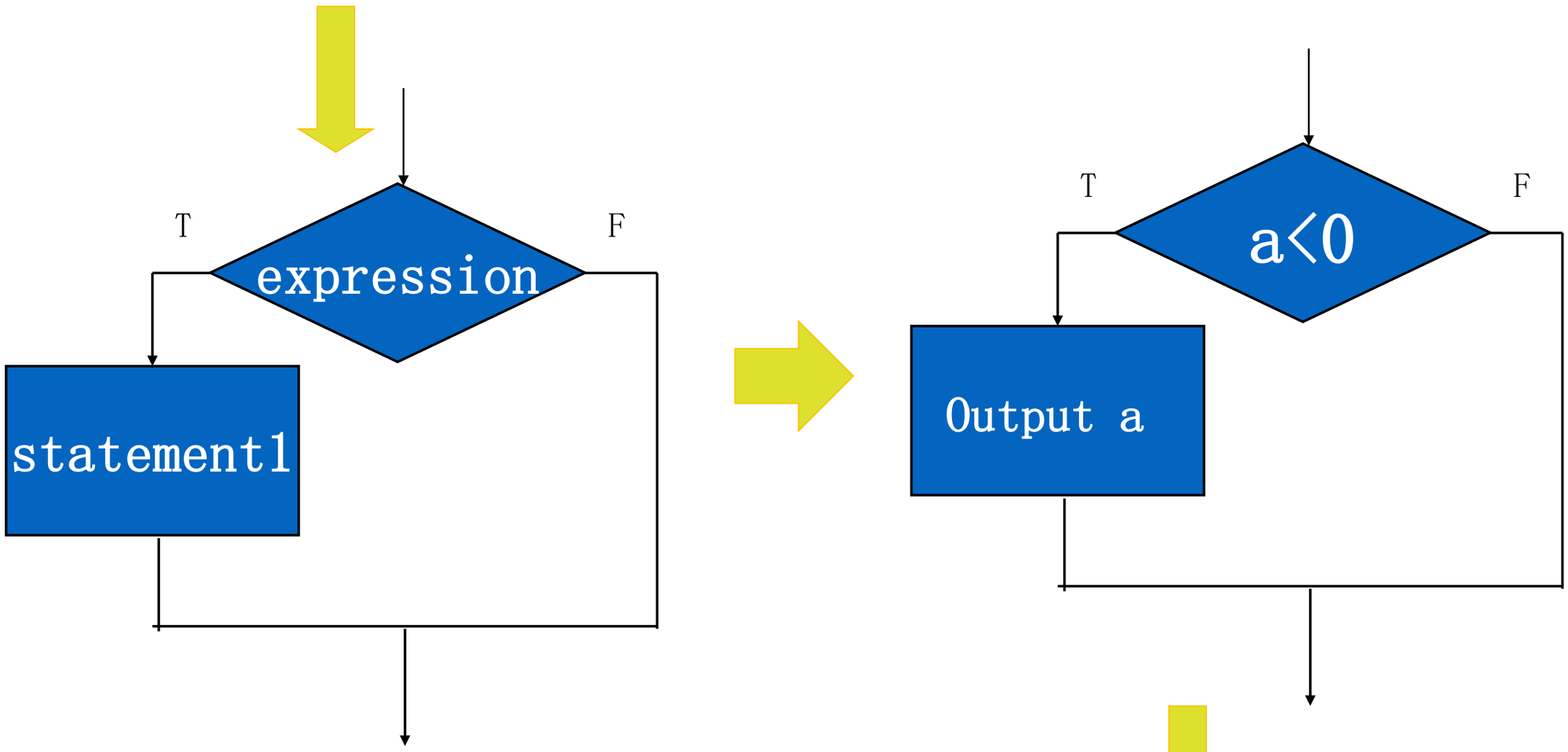
Single-alternative selection structure

- If Statement
  - Tests whether a condition is true
  - Then executes a statement if it is true
  - Does nothing if it is false

Syntax: `if (condition) statement`

Single-alternative selection structure

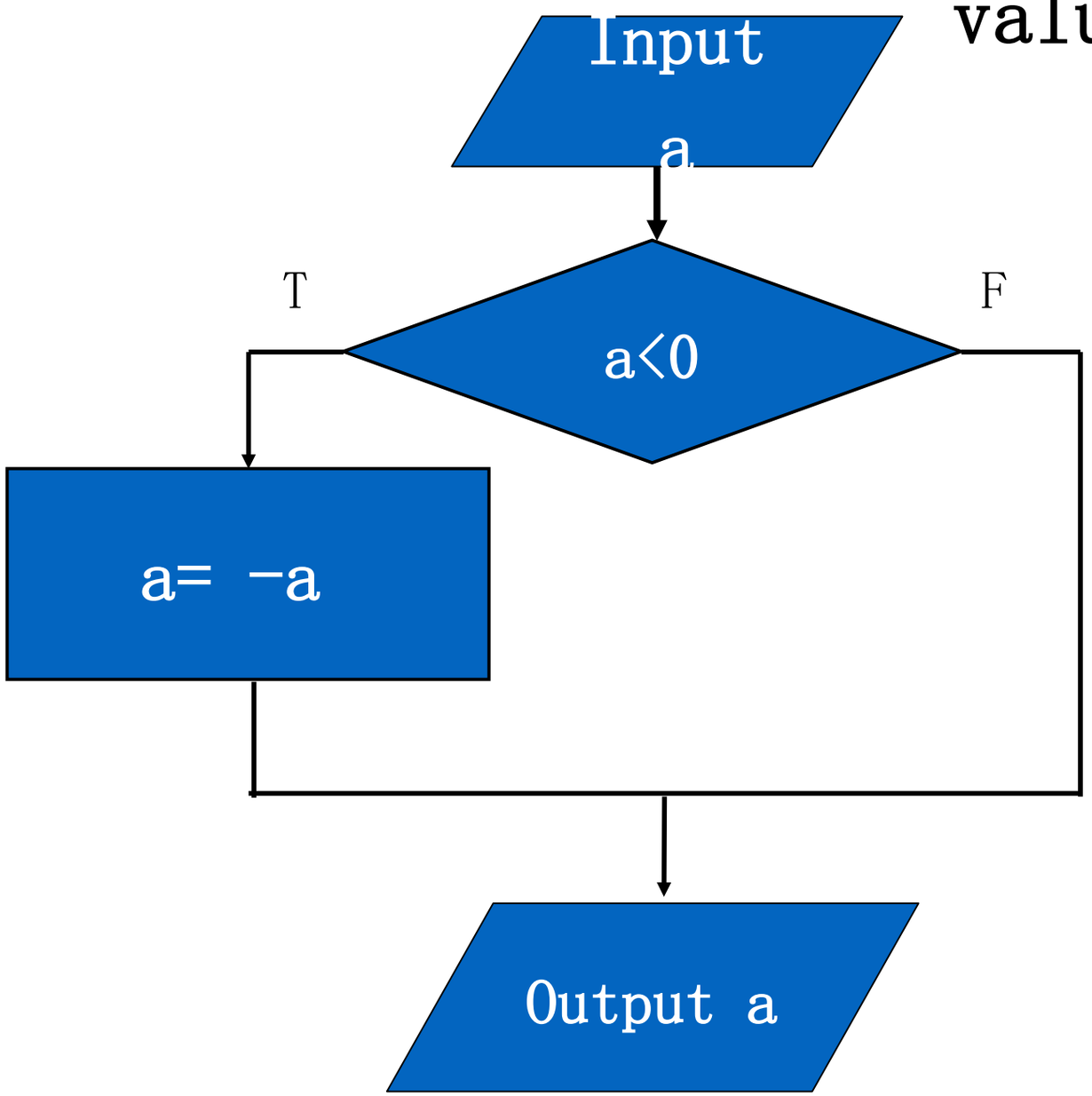
if (expression) <statement 1>



if (a<0) printf("%d",a);

Single-alternative selection structure

【 ex.1】Input number a, output it with a positive value.



scanf("%d",&a);

if(a<0) a=-a;

printf("%d",a);

Single-alternative selection structure

```
void  main()
{
 int a;
 scanf("%d",&a);
 if(a<0)  a=-a;
 printf("%d",a);
}
```

result:

Input: 5

Output:5

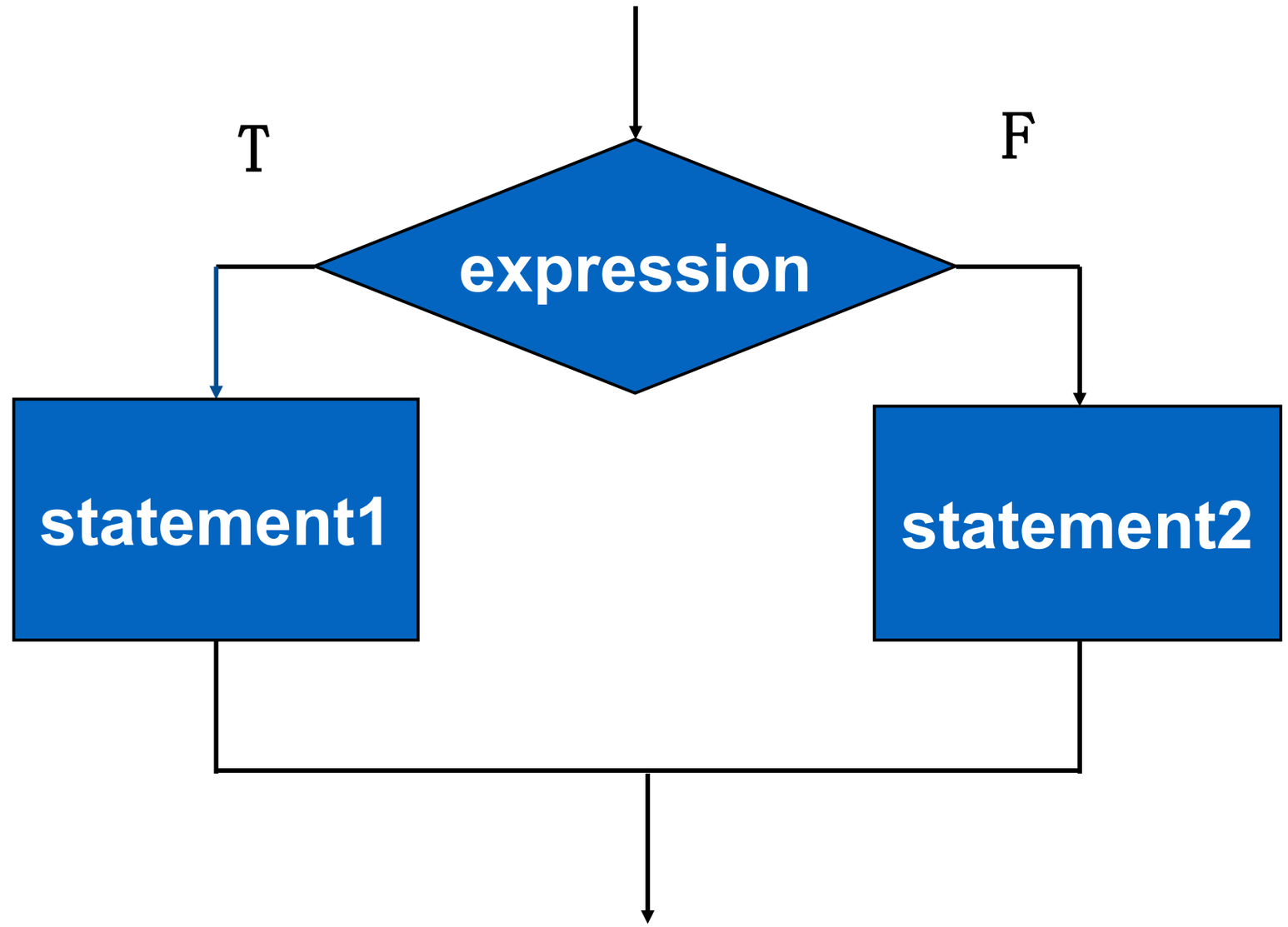
Input: -6

Output:6

Dual-alternative selection structure

- The dual-alternative selection structure evaluates a condition and executes one of **two possible execution paths**.
- If the condition is true, then one block of code is executed. If the condition is false, the other block of code is executed.

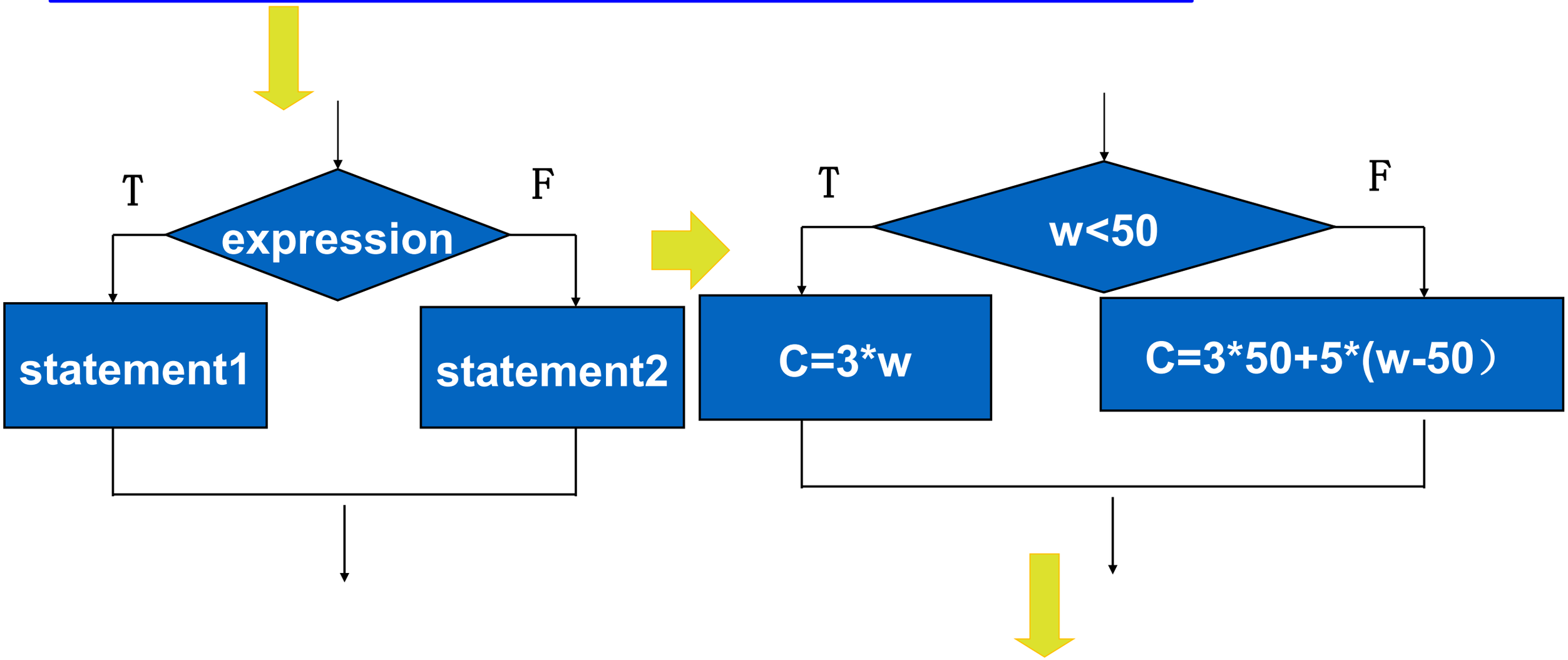
Dual-alternative selection structure



An example of a dual-alternative selection structure is the if-else statement.

Dual-alternative selection structure

```
if  (expression)     <statement 1>
else    <statement 2>
```




```
if  (w<50)   c=3*w;
else   c=3*50+5*(w-50);
```

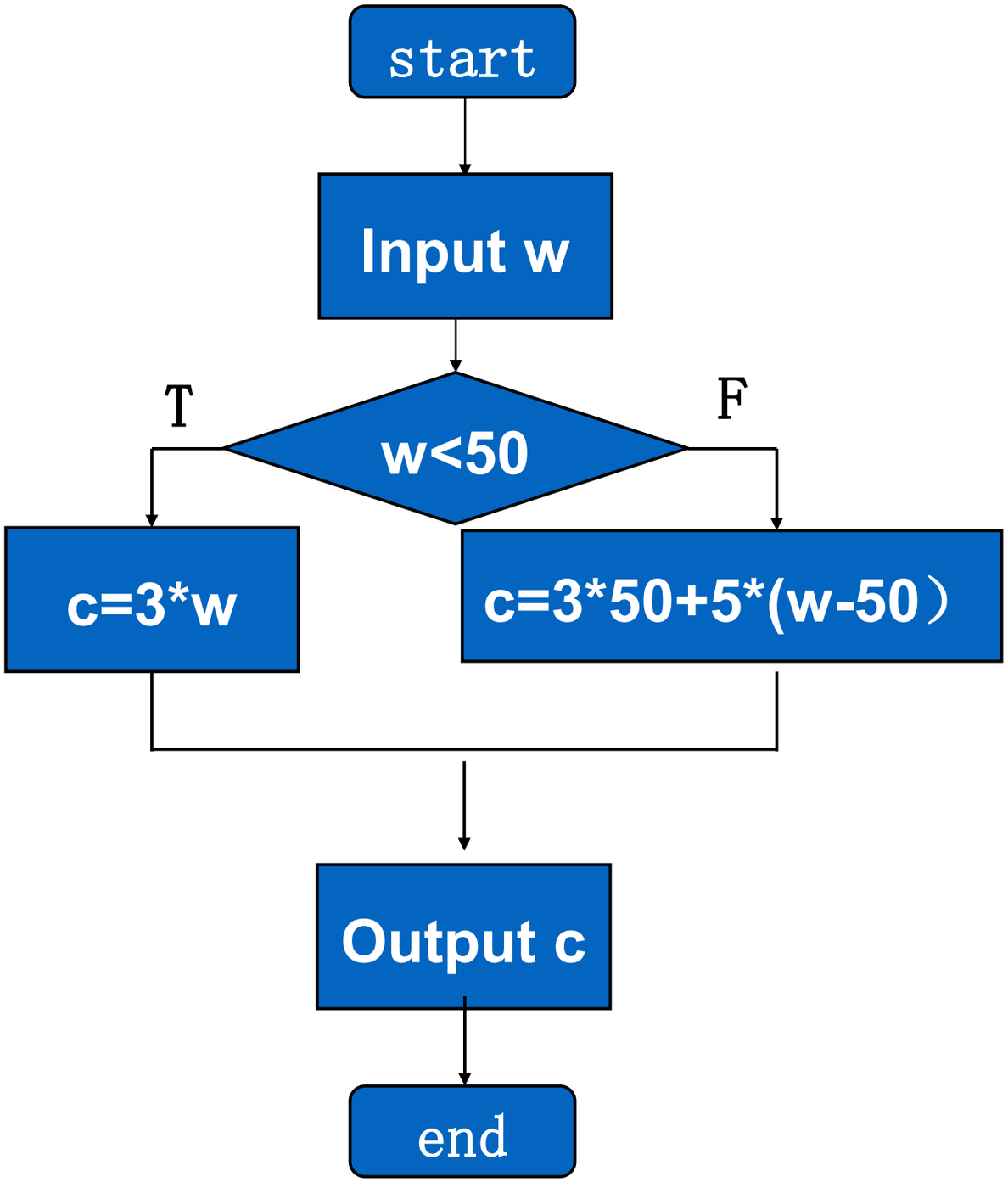
Dual-alternative selection structure

[ex.2] Compute the cost of baggage carried by train. Input the weight of the baggage, compute the cost.

```
standard of cost:
  (1) lighter than  50 kg(include 50 kg), $ 3 /kg
  (2) over weight than 50 kg, 50 kg like (1), the
other excess, $5 /kg
```

analysis cost	**not overweight**	**3*weight**
	overweight	**3*50+5*(weight-50）**

Dual-alternative selection structure




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

```
if(w<0) c=3*w;
else c=3*50+5*(w-50)
```

```
printf("%d",c);
```

Dual-alternative selection structure

```
void  main()
{
 int w,c;
 scanf("%d",&w);

 if(w<0) c=3*w;

 else c=3*50+5*(w-50)
 printf("%d",c);
}
```

result:

Input: 5

Output:15

Input: 55

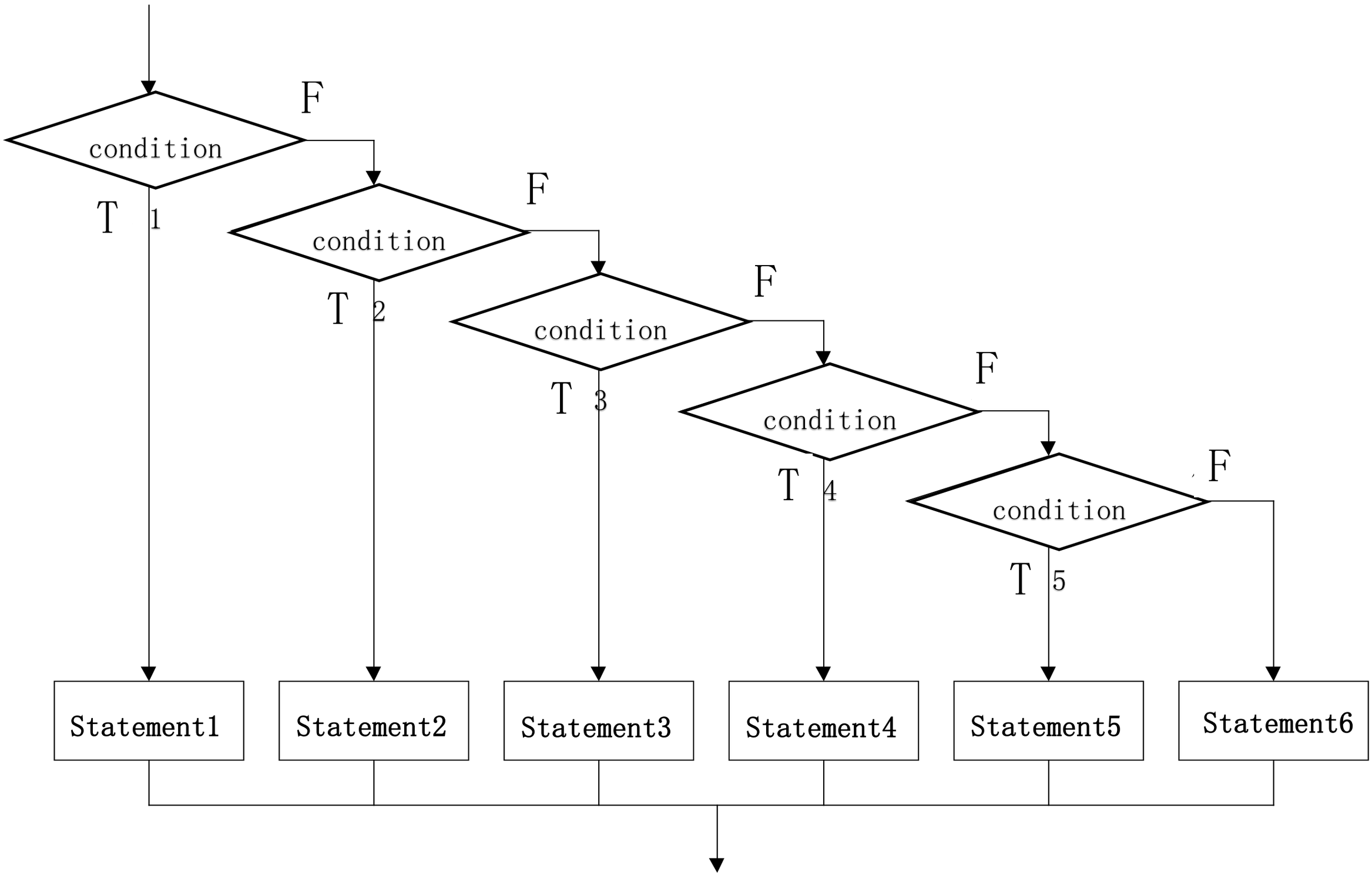
Output:175

multiple-branching construction

- if else nesting

if(condition 1) statement 1
else if(condition 2) statement 2
else if(condition 3) statement 3
. . .
else if(condition n) statement n
else statement n+1 ;

multiple-branching construction



Example-Grade Values

- A professor generates letter grades using the following table.
- Goal: given the score, to print the grade for the student.

Score	Grade
0-60	F
61-70	D
71-80	C
81-90	B
91-100	A

```
int main()
{        ...
    printf("F\n");
    printf("D\n");
    printf("C\n");
    printf("B\n");
    printf("A\n");
}
```

Example-Grade Values

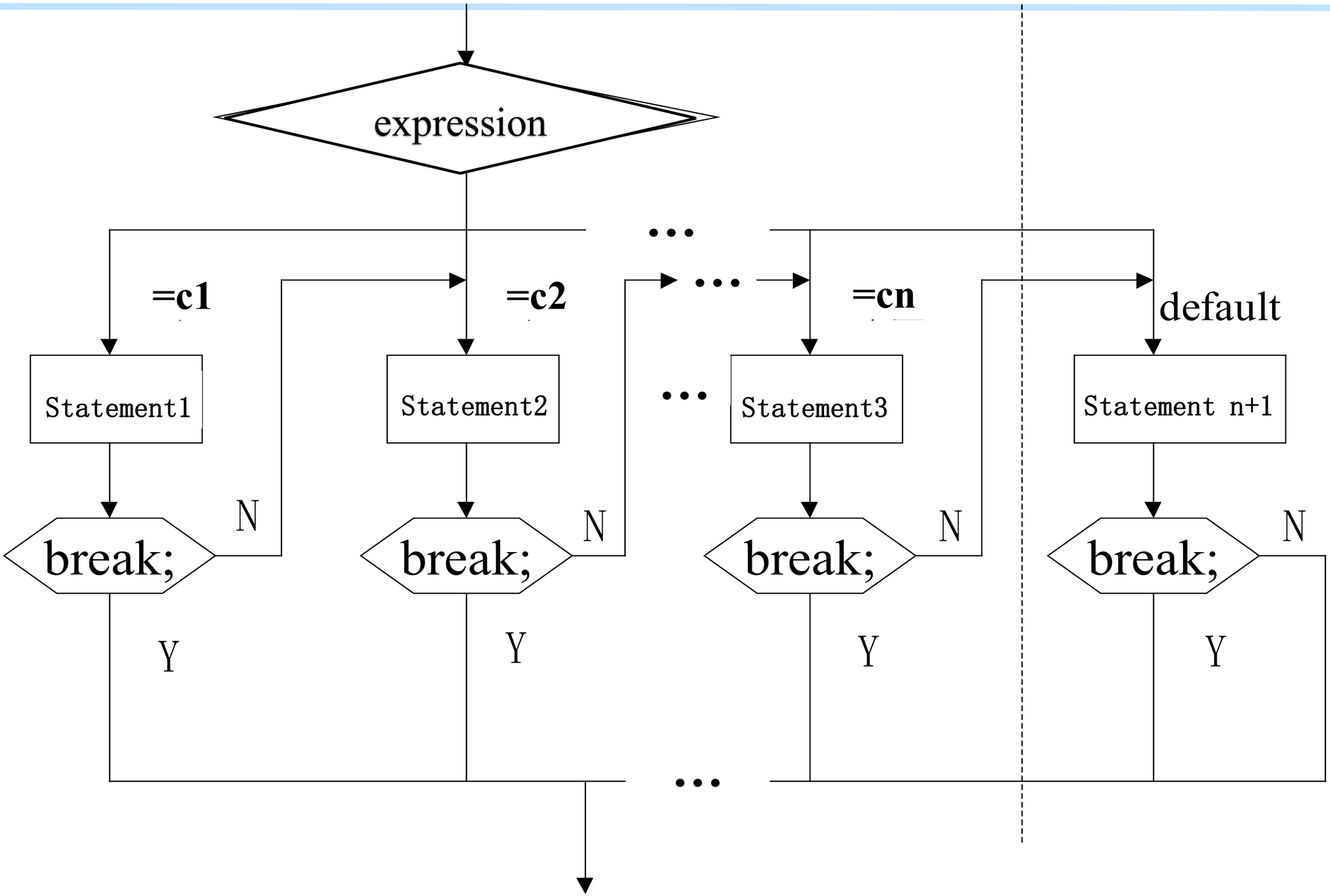
```
#include "stdio.h"
void main()
  { int score;char level;
    printf("input score=");scanf("%d",&score);
    if(score<0||score>100) { printf("Invalid input.\n");return; }
   if(score<60) level='E';
     else if(score<70) level='D';
        else if(score<80) level='C';
              else if(score<90) level='B';
                   else level='A';
  printf("level is %c\n",level);
}
```

multiple-branching construction

Switch statement

```
switch( expression)
    { case constant-expression_1: statement 1; [break;]
      case constant-expression_2: statement 2; [break;]
            .
            .
            .
      case constant-expression_n: statement n; [break;]
      [default: statement n+1; [break;] ]
}
```

multiple-branching construction



multiple-branching construction

```
#include "stdio.h"
void main()
  { int score;char level;
    printf("input score="); scanf("%d",&score);
    if(score<0||score>100) { printf("Invalid input.\n");return; }
    switch(score/10)
    {  case 9:  case 10:level= 'A';break;
       case 8:level= 'B';break;
       case 7:level= 'C';break;
       case 6:level= 'D';break;
       default:level= 'E';
    }
   printf("level is %c\n",level);
}
```

multiple-branching construction

- What's the output of the following code:

```
#include<stdio.h>
int main()
{ char n = 'c';
switch(n++)
  { default: printf("Error"); break;
    case 'a': printf("good "); break;
    case 'c': printf("morning ");
    case 'd': printf("class");
  }
```

What's the output of the following code:

```
#include <stdio.h>
void main()
{       char mark;
        printf("\nPlease input mark:");
        scanf("%c",&mark); //input marks
        switch(mark)
        {       case 'A':printf("90~100\n");
         case 'B':printf("80~89\n");
         case 'C':printf("70~79\n");
         case 'D':printf("60~69\n");
         case 'E':printf("0~59\n");
         default:printf("Error\n"); }
}
```

$$\begin{cases} 100 \geq mark \geq 90 & A \\ 80 \leq mark < 90 & B \\ 70 \leq mark < 80 & C \\ 60 \leq mark < 70 & D \\ mark < 60 & E \end{cases}$$

```
Please input mark:B
80~89
70~79
60~69
0~59
Error
Press any key to continue_
```

What's the output of the following code:

```
#include <stdio.h>
void main()
{       char mark;
        printf("\nPlease input mark:");
        scanf("%c",&mark); //input marks
        switch(mark)
        {       case 'A':printf("90~100\n"); break;
         case 'B':printf("80~89\n");break;
         case 'C':printf("70~79\n"); break;
         case 'D':printf("60~69\n"); break;
         case 'E':printf("0~59\n"); break;
         default:printf("Error\n"); }
}
```

$$\begin{cases} 100 \geq mark \geq 90 & A \\ 80 \leq mark < 90 & B \\ 70 \leq mark < 80 & C \\ 60 \leq mark < 70 & D \\ mark < 60 & E \end{cases}$$

```
Please input mark:B
80~89
Press any key to continue
```

Nested If Statements

- If statements are still statements
- So they can be put in compound statements
- Or they can be used inside another if statement:

```
if (year == 2016)
    if (month == january)
    rent *= 1.01;
else
    if (month == january)
    rent *= 1.02;
```

What does this do?

Classic Mistake No. 1

- Else *always* pairs with the nearest if
- Ignoring all indentation
- In this case
  - else pairs with if (month == january)
  - only executes if month != january
  - the second if statement is *never* true
  - and the rent never increases by 2%

Solution

- Always, always use compound statements
- Add comment so you can see the logic

```
if (year == 2016)
  { // 2016
    if (month == january)
      { // january, 2016
      rent *= 1.01;
      } // january, 2016
  } // 2016
else
  { // not 2016
   if (month == january)
      { // january, not 2016
      rent *= 1.02;
      } // january, not 2016
  } // not 2016
```

Classic Mistake No. 2

```
year = 2017;
if (year = 2016)
  printf("2016");
else
  printf("not 2016");
```

- We missed an equals sign
- We used the assignment operator
- Instead of the comparison operator
- So we assigned 2016 to year
- Then tested it (2016 is not 0, so it is true)

Ternary If Operator

- One of the weirdest parts of C

  condition ? expression1 : expression2

- An operator with *two* symbols
  - And *three* operands
- Shortcut for an if statement
  - but returns an assignable value

Sample Code

```
if (x>y)
        max = x;
else
        max = y;
```

With ternary operator:

```
max = (x>y)? x:y;
```

variable = condition ? expression1 : expression2

Some value to be expression 1 or expression 2

Examples

[ex.2]output the value of the function f (x) with an input x.

$$f(x)=\begin{cases} x & x<1 \\ 2x-1 & 1\le x<10 \\ x^2+2x+2 & x\ge 10 \end{cases}$$

Examples

```
#include <stdio.h>
void main()
{    float x,y;
     printf("\nPlease input x:");
     scanf("%f",&x);   //input x
     if (x>=10)
        y=2*x-1;
     if (x>=1&&x<10)
        y=x*x+2*x+2;
     if (x<1)
        y=x;
     printf("y=%f\n",y);
}
```

$$f(x)=\begin{cases} x & x<1 \\ 2x-1 & 1\le x<10 \\ x^2+2x+2 & x\ge 10 \end{cases}$$

Examples

```
#include <stdio.h>
void main()
{    float x,y;
     printf("\nPlease input x:");
     scanf("%f",&x);   //input x
     if (x>=1)
      if (x<10)
          y=2*x-1;
      else
          y=x*x+2*x+2;
      else
         y=x;
     printf("y=%f\n",y);
}
```

$$f(x)=\begin{cases} x & x<1 \\ 2x-1 & 1\le x<10 \\ x^2+2x+2 & x\ge 10 \end{cases}$$

Exercises1

Given a function:

$$f(x)=\begin{cases} 2x & x<10 \\ 3x-10 & 10 <= x<20 \\ 5x-100 & x>=20 \end{cases}$$

Write a program to accept the user input for x, and display the value of y

Exercises 2

- 企业放发的奖金根据利润提成。设企业的利润为I，提成标准如下：
- I＜10万元时，提成10%;
- 10万元＜I≤20万元时,低于10万元部分仍按10％提成，高于10万元部分按7.5％提成；
- 20万元＜I≤40万元时,低于20万元部分按前面方法提成，高于20万元部分按5％提成；
- 40万元＜I≤60万元时,低于40万元部分按前面方法提成，高于40万元部分按3％提成；
- 60万元＜I≤100万元时,低于60万部分按前面方法提成，高于60万部分按1.5％提成；
- 100万元＜I时，低于100万元部分按前面方法提成，高于100万元部分按1％提成。
- 编程输入利润I，计算输出提成金额。