CSC10001 – Introduction to Programming

4th lecture: Control structure: Repetition

fit hcmus
Instructor: TRAN Thi Thao Nhi

October 24, 2023

Increment/decrement operators

++ and -- are operators that add and subtract 1 from their operands

```
num = num + 1;

num += 1;
num++; // postfix mode
++num; // prefix mode
```

```
num = num - 1;

num -= 1;
num--; // postfix mode
--num; // prefix mode
```

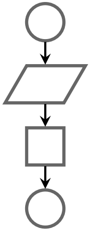
```
cout << num++;
cout << ++num;

// in mathematical expressions
int a = 2, b = 5, c;
c = a * b++; // c = a * ++b;
cout << a << " " << b << " " << c;

// in relational expressions
if (c++ > 10)
    cout << c;
```

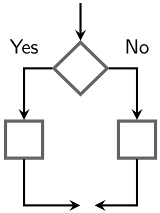
Control structures

A computer can process a program in one of the following ways



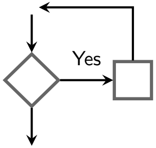
Sequence structure

- follows the statements in order



Selection struture

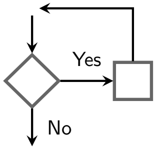
- executes particular statements depending on condition(s)



Repetition structure

- repeats particular statements a certain number of times based on condition(s)

Repetition structure



A repetition structure *(loop)* is a control structure that causes a statement or group of statements to repeat

- know number of iterations
- "until" something happens

```
// add 5 numbers to find the
    average
int n1, n2, n3, n4, n5;
cin >> n1 >> n2 >> n3 >> n4 >> n5;
int sum_5numbers = n1 + n2 + n3 +
    n4 + n5;
int avg = sum_5numbers / 5;



// find sum of the first n (n <
    100) numbers, from 1 to n
int n, sum;
cin >> n;
if (n >= 1) sum += 1;
if (n >= 2) sum += 2;
...
if (n >= 98) sum += 98;
if (n >= 99) sum += 99;
// => any mistake?
// another mathematical solution?
```

`while` loop statement

```
while (expression)
    statement{s};
```

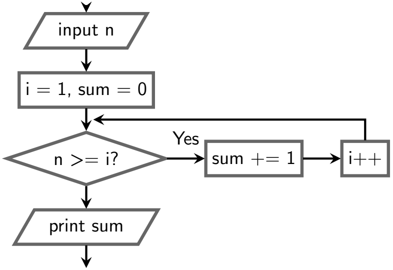
- `expression`: (loop header/condition) a boolean expression
- `statement{s}`: (loop body) a statement or a block of statements with `{}`
- while the expression is `true`, the statements is executed; this cycle repeats until the expression is `false`

Parameterization: divive task into unchanged part `S` and changed part `i` (parameter) $\rightarrow$ each specific value of `i` results in a specific `S(i)`
$\Rightarrow$ to form a loop, use *loop variable(s)* for the parameter and repeat the unchanged part based on the loop variable

Parameterization

```cpp
// find sum of the first n (n < 100) numbers
int n, sum;
cin >> n;
if (n >= 1) sum += 1;
if (n >= 2) sum += 2;
...
if (n >= 98) sum += 98;
if (n >= 99) sum += 99;
cout << "sum of the first n numbers = " << sum;
```

⇒ which is the changed part? unchanged part?



```cpp
int n, sum = 0, i = 1;
cin >> n;

while (n >= i) {
     sum += i;
     i++;
}
cout << "sum of the first n
     numbers = " << sum;
```

TODO: find average of 5 numbers, 5.35 (flowchart)

Counter

A counter is a variable that is regularly incremented or decremented each time a loop iterates

```
int n, sum = 0, i = 1; // i is a counter
cin >> n;

while (i <= n) {
    sum += i;
    i++;
}
cout << "sum of the first n numbers = " << sum;
```

what happen if

- the expression is false at the beginning? → pretest loop
- the expression never meet the `false` value (terminate condition)? → infinite loop
- forget the braces in block statements?
- forget initialize counter/loop variable?

Design while loop

Counter-controlled while loop: know exactly how many times certain statements need to be executed

```
// find sum of the first n numbers
while (counter < limit)
```

Sentinel-controlled while loop: know the last entry is a special value (sentinel)

```
// find the average of positive numbers, stop when
    receiving a negative number (-1)
while (counter != sentinel)
```

Flag-countrolled while loop: (conditional loop) executes as long as a particular condition exists

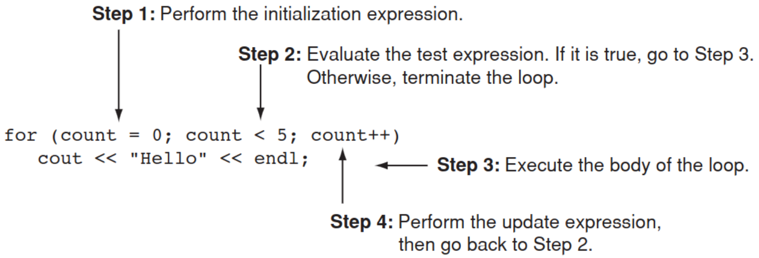
```
// number guessing game: guess a integer number until
    it is correct
// after each turn, print a clue whether the guessed
    number is lower or higher than the answer
while (boolean_variable)
```

TODO: [programming challenges] 2, 3 (flowchart)

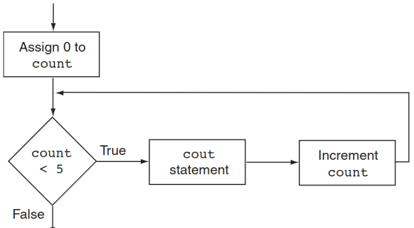
for loop

counter-controlled while loop

```
for (initial statement; loop condition; update
    statement)
     statement{s}
```


Step 1: Perform the initialization expression.
Step 2: Evaluate the test expression. If it is true, go to Step 3. Otherwise, terminate the loop.
Step 3: Execute the body of the loop.
Step 4: Perform the update expression, then go back to Step 2.

```
for (count = 0; count < 5; count++)
    cout << "Hello" << endl;
```



```
for (float i = 0; i < 5; i++)
    statement{s}

for (int i = 0; i < 5; i++);

for ( ; ; )
    statement{s}
```

do-while loop

The do-while loop is a posttest loop, which means its expression is tested after each iteration

```
do
    statement{s}
while (expression);
```

Determine whether input number is positive or negative, stop when receiving zero value

```
int n;
do {
    cout << "input int n: ";
    cin >> n;
    cout << "n is " << (n > 0 ? "positive": (n < 0 ? "negative" :
    "zero")) << "\n";
} while (n != 0);
```

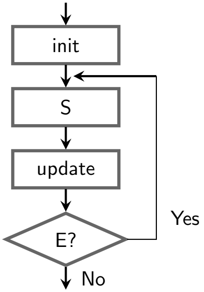
```
int n;
while (n != 0) {
    cout << "input int n: ";
    cin >> n;
    cout << "n is " << (n > 0 ? "positive": (n < 0 ? "negative" :
    "zero")) << "\n";
}
```

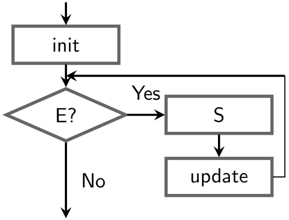
do-while, while, for

```
<init>
do {
    <S>;
    <update>;
} while (<E>);
```

```
<init>
while (<E>) {
    <S>;
    <update>;
}
```

```
for (<init>; <E>; <update>) {
    <S>;
}
```





TODO: [DS.Malik book - Programming exercises] 7

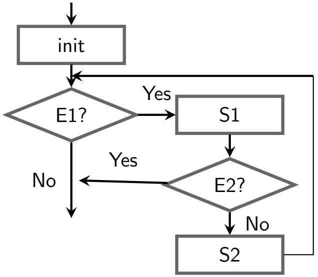
break and continue statements

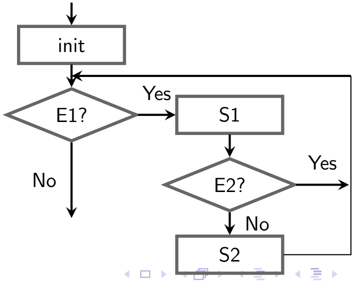
The break statement causes a loop to terminate early.

The continue statement causes a loop to stop its current iteration and begin the next one

```
while (<E1>) {
    <S1>;
    if (<E2>)
        break;
    <S2>;
}
```

```
while (<E1>) {
    <S1>;
    if (<E2>)
        continue;
    <S2>;
}
```





Nested loops

A loop that is inside another loop is called a nested loop

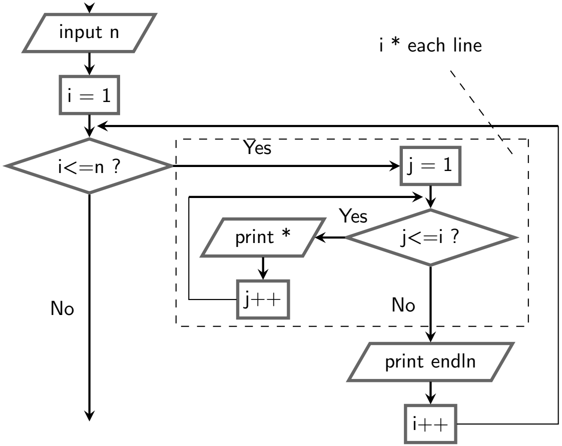
TODO: Print a right triangle with size of n

```
*
**
***
****
*****
```

TODO: Determine whether input number is prime, stop when receiving zero value?

Pseudo-infinite loop

```
while (1) {
    <S>;
    if (<E>)
        break;
}
```



Flattening a nested loop

TODO: Input x, n. Calculate

$$S = 1 + x^2 + ... + x^n = \sum_{i=0}^{n} x^i$$

```
int x, n, s = 0;
cout << "x = "; cin >> x;
cout << "n = "; cin >> n;

for (int i = 0; i <= n; i
    ++) {
    // calculate t = x^i
    int t = 1;
    for (int j = 1; j <= i;
     j++) {
        t *= x;
    }
    s += t;
}
```

```
int x, n;
cout << "x = "; cin >> x;
cout << "n = "; cin >> n;

int t = 1, s = t;
for (int i = 1; i <= n; i++)
     {
    t *= x; // t = x^(i-1)*x
    s += t;
}
```

TODO

- Finish chapter 5
- Read chapter 6