PROGRAMMAZIONE PROCEDURALE

A.A. 2021/2022
CONDITIONAL STATEMENTS
AND LOOPS EXAMPLES
WHY LOOPS ARE IMPORTANT

- Loop can be defined as one of the basic logical structures of computer programming.

- Defining loops allows computers to perform the same task repeatedly.

- Every time you might use the words "each" and "every" in the natural language description of your program's specification you want a loop.

- Also forever
  ✓ while(1)
EXAMPLES ON LOOPS
Logic Conditions

int x=1;

while(x=1) {
    //...
    printf ("Insert 1 to continue");
    scanf("%d", &x);
}

If you use a single equal sign to check equality, your program will instead assign the value on the right side of the expression to the variable on the left hand side, and the result of this statement is the value assigned. In this case, the value is always 1, which is treated as always true.

int x=1;

while(x==1) {
    //...
    printf ("Insert 1 to continue");
    scanf("%d", &x);
}
int n = 4;

for (int i = 0; i < n; i++) {
    printf("OK\n");
}

int n = 4;
int i = 0;

while (i < n) {
    printf("OK\n");
    i++;
}

int n = 4;

do {
    printf("OK\n");
    i++;
} while (i < n);
DO...WHILE

#include <stdio.h>

int main()
{
    double number, sum = 0;

    // loop body is executed at least once
    do {
        printf("Enter a number: ");
        scanf("%lf", &number);
        sum += number;
    } while(number != 0.0);

    printf("Sum = %.2lf", sum);
    return 0;
}
// Arrays example
#include <stdlib.h>

int main () {
    int foo [] = {16, 2, 77, 40, 12071};
    int n, result=0;

    for ( n=0 ; n<5 ; ++n ) {
        result += foo[n];
    }

    printf("%d", result);
    return 0;
}
```c
const int width= 5;
const int height= 3;

int main () {

    int mat [height][width];

    for (int n=0; n< height; n++)
        for (int m=0; m< width; m++)
            mat[n][m]= (n+1)*(m+1);

}
```

![Matrix Table]

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
</tr>
</tbody>
</table>
When the name of an array appears as a function argument, the compiler implicitly converts it into a pointer to the array’s first element.

✓ `int name[]` or
✓ `int *name` is the same.
#include <stdio.h>

void fun(int a[], int n) {
    for (int i = 0; i < n; ++i )
        a[i] = i;
}

int main(){
    int a[3];
    fun(a, 3);
}

#include <stdio.h>

void fun(int* a, int n) {
    for (int i = 0; i < n; ++i )
        *(a + i) = i;
}

int main(){
    int a[3];
    fun(a, 3);
}

**C does not have array** variables

It is really just working with pointers with an alternative syntax.
#include <stdio.h>

int matrix_sum (int row, int column, int m[row][column]) {
    int sum = 0;
    
    for (int i = 0; i < row; i++)
        for (int j = 0; j < column; j++)
            sum += m[i][j];
    return sum;
}

int main()
{
    int m[2][3]= {1,4,5,2,1,1};
    int result;
    result= matrix_sum(2,3, m);
    printf("%d", result);
}
AN ADVANCED EXAMPLE
#include <stdio.h>

int main() {
    int n, c, d, swap;
    printf("Enter number of elements\n");
    scanf("%d", &n);
    printf("Enter %d integers\n", n);
    int array[n];

    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);

    for (c = 0; c < (n - 1); c++)
    {
        for (d = 0; d < n - c - 1; d++)
        {
            if (array[d] > array[d+1])
            {
                swap = array[d];
                array[d] = array[d+1];
                array[d+1] = swap;
            }
        }
    }
    return 0;
}
BUBBLE SORT

6 5 3 1 8 7 2 4
EXAMPLES ON SELECTION STATEMENTS
LOGIC CONDITIONS

int age= 17;
if (age = 18)
printf(“OK!”);

What is the value of age = 18? 18
Is 18 true of false in C? true
It always prints “OK!” and age becomes 18

int age= 17;
if (age == 18)
printf(“OK!”);

It prints “OK!” only when age is equal to 18
age remains equal to 17
#include <stdio.h>

int main()
{
    int number = 0;
    printf("Enter an integer: ");
    scanf("%d", &number);

    // True if remainder is 0
    if (number % 2 == 0)
    {
        printf("%d is an even integer.", number);
    }
    else
    {
        printf("%d is an odd integer.", number);
    }

    return 0;
}
CASCADE OF IFS

#include <stdio.h>

int main(){
    char grade= 'F';
    int score= 0;
    printf("Enter a score: ");
    scanf("%d", &score);

    // True if remainder is 0
    if (score >= 90)
        grade= 'A';
    else if (score >= 80)
        grade= 'B';
    else if (score >= 70)
        grade= 'C';
    else if (score >= 60)
        grade= 'D' ;

    printf("Your grade is %c: ", grade);

    return 0;
}

#include <stdio.h>

int main(){
    char grade= 'F';
    int score= 0;
    printf("Enter a score: ");
    scanf("%d", &score);

    // True if remainder is 0
    if (score >= 90)
        grade= 'A';
    else
        if (score >= 80)
            grade= 'B';
        else
            if (score >= 70)
                grade= 'C';
            else
                if (score >= 60)
                    grade= 'D' ;

    printf("Your grade is %c: ", grade);

    return 0;
}
if (x > 10) {
    if (x < 20)
        printf("x is between 10 and 20");
    else
        printf("x is greater than 20");
}

if (x > 10) {
    if (x > 20)
        printf("x is between 10 and 20");
    else
        printf("x is greater than 20");
}

if (x > 10) {
    if (x < 20)
        printf("x is between 10 and 20");
    else
        printf("x is greater than 20");
}
COMMON ERRORS

if (age >= 18);
  printf("You can drink!\n");
  This will always print “Ok” even if age is less than 18

if (age >= 18)
  printf("You can drink!\n");
  printf("Age greater than 18\n");
  This will always print “Age greater than 18” even if age is less than 18

if( a < b)
  { do something here; }
else if( a > b)
  { do this other thing; }
else (a == b)
  { do something different; }

Else is not allowed to have a condition
COMMON ERRORS

int x = 15
if (20 > x > 10)
    printf("x is between 10 and 20");

> is associative left-to-right…

20 > x true(1)
1 > 10 false(0)

if (x > 10 && x < 20) OK!
    printf("x is between 10 and 20");

1. If you have multiple cases, an if for each
2. If you condition is long, decompose and use logical operators
INFINITE LOOPS

#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i = 1;
    while (1) {
        i++;
        if (i < 0) {
            printf("%d\n", i);
            break;
        }
    }
}

MacBook-Francesco:Programmi francescosantini$ ./a.out
-2147483648
An int in two’s complement

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>01111111 11111111 11111111 11111111</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>01111111 11111111 11111111 11111110</td>
<td>2,147,483,646</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>00000000 00000000 00000000 00000010</td>
<td>2</td>
</tr>
<tr>
<td>00000000 00000000 00000000 00000001</td>
<td>1</td>
</tr>
<tr>
<td>00000000 00000000 00000000 00000000</td>
<td>0</td>
</tr>
<tr>
<td>10000000 00000000 00000000 00000000</td>
<td>-2,147,483,648</td>
</tr>
<tr>
<td>10000000 00000000 00000000 00000000</td>
<td>-2,147,483,647</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>11111111 11111111 11111111 11111110</td>
<td>-2</td>
</tr>
<tr>
<td>11111111 11111111 11111111 11111111</td>
<td>-1</td>
</tr>
</tbody>
</table>