

# C: an introduction

Control flow: Conditionals

1

## Program: building blocks

- Variables
  - Store data (input, intermediate values, results)
- Expressions
  - Manipulate variables
- Control structures
  - Make decisions (if) or repeat (for, while) statements
- Functions
  - Combine expressions and structures for parameterization and re-use

2

## control flow

- Two main aspects of a program:
  - the *actions* to be executed;
  - the *order* of these actions (*flow of control*)
- what can be done in C:
  - sequencing `func1(); func2();`
  - branching/selection `if (a) func1();`
  - Iteration/looping `while (i<10)  
    func1();`
  - functions `func1(a, b);`
  - recursion `func(int i)  
    {func(i-1);}`

## if statement

```
if (condition) {  
    statement;  
}
```

- If `<condition>` evaluates to TRUE, then the following `<statement>` is executed.
  - `<condition>` is anything that evaluates to TRUE (not 0) or FALSE (0).
  - `<statement>` can be a simple or compound statement (block {})
- Parentheses around the condition are required
- Indentation not needed for compiler
  - Needed for readability

# What are True and False?

- An expression that is logically TRUE (using the relational operators) will evaluate to 1

```
c = (a <= b);
```

- an (arithmetic) expression that is NON-ZERO is interpreted as logically TRUE (It does not have to be a Boolean expression. It can be any expression that evaluates to a number, in which case zero is false and all other numbers are true.)

```
if (b - a) { ... }
```

## conditional\_1.c

```
1 #include <stdio.h>
2
3 /*
4 conditional_1.c
5 reads two ints-praises for following directions
6 taken from https://www.cs.umd.edu/class/fall2015/cmsc106
7 */
8
9 int main() {
10 int x, y;
11
12 x = 4;
13 y = 4;
14
15 printf("input x = %d, y = %d \n", x, y);
16
17 if (x == y && x > 0){
18     printf("Good Job\n");
19     printf("We are done here\n");
20 }
21
22 if (x == y && x < 0)
23     printf("Good Job\n");
24     printf("We are done here\n");
25
26 return 0;
27 }
```

```
frankvp@CRD-L-08004:~/Controlflow$ gcc conditional_1.c -o conditional_1
frankvp@CRD-L-08004:~/Controlflow$ ./conditional_1
input x = 4, y = 4
Good Job
We are done here
We are done here
frankvp@CRD-L-08004:~/Controlflow$
```

```

1/*****
2conditional_2.c
3Author: Ben Humphrey digiben@gametutorials.com
4Program: Questions
5Description: Asks/Answers questions using if/else statements.
6Date: 5/18/00
7© 2000-2003 GameTutorials
8*****/
9#include <stdio.h>
10
11void main()
12{
13    int age=0;
14
15    printf("How old are you? ");
16    scanf("%d", &age);
17
18    if(age > 20)
19    {
20        printf("You're over 20 huh?\n");
21    }
22
23    if(age > 30)
24        printf("You're over 30!\n");
25
26    if(age < 20) {
27        printf("You're a young'n!\n");
28    }
29
30    if(age < 20 && age > 12)
31    {
32        printf("Being in your teens can be tough...\n");
33    }
34
35    if(age == 100)
36        printf("WOW! What's your secret!\n");
37
38}
39
40
41

```

## • conditional\_2.c

```

frankvp@CRD-L-08004:../Controlflow$ gcc conditional_2.c -o conditional_2
frankvp@CRD-L-08004:../Controlflow$ ./conditional_2
How old are you? 28
You're over 20 huh?
frankvp@CRD-L-08004:../Controlflow$ ./conditional_2
How old are you? 61
You're over 20 huh?
You're over 30!?
frankvp@CRD-L-08004:../Controlflow$ ./conditional_2
How old are you? 101
You're over 20 huh?
You're over 30!?
frankvp@CRD-L-08004:../Controlflow$ ./conditional_2
How old are you? 100
You're over 20 huh?
You're over 30!?
WOW! What's your secret!?
frankvp@CRD-L-08004:../Controlflow$

```

KU LEUVEN

8

```

1/* head_or_tail.c
2based on Todd RPI
3The C library function int rand(void) returns a
4pseudo-random number in the range of 0 to RAND_MAX.
5*/
6
7#include <stdio.h>
8#include <stdlib.h>
9#include <time.h>
10
11
12int main(void)
13{
14    int coin, choice;
15
16    /* Intializes random number generator */
17    /* The time() function returns information about the current
18time of day, a value that's constantly changing. The NULL
19argument helps solve some problems, but time() returns an
20ever-changing value */
21
22    srand(time(NULL));
23    coin = rand() % 2;
24
25    printf(" Make your choice (0: head - 1: tail):");
26    scanf(" %d", &choice);
27
28    if (choice == coin)
29    {
30        printf(" well done! \n");
31    }
32
33    return 0;
34}

```

## head\_or\_tail.c

```

frankvp@CRD-L-08004:../Controlflow$ gcc head_or_tail.c -o head_or_tail
frankvp@CRD-L-08004:../Controlflow$ ./head_or_tail
Make your choice (0: head - 1: tail):1
frankvp@CRD-L-08004:../Controlflow$ ./head_or_tail
Make your choice (0: head - 1: tail):1
frankvp@CRD-L-08004:../Controlflow$ ./head_or_tail
Make your choice (0: head - 1: tail):1
frankvp@CRD-L-08004:../Controlflow$ ./head_or_tail
Make your choice (0: head - 1: tail):1
well done!
frankvp@CRD-L-08004:../Controlflow$

```

9

## Statements and Blocks

- Simple, single statement:

```
x = a + b;
```

- Compound statement (sequential structure) is created by grouping simple statements into a *block* by enclosing them in the braces:

```
{  
    x = a + b;  
    y = sin(x);  
}
```

- Compound statements are syntactically equivalent to a single statement.
- Block can be empty {}

## Nested `if` statements

- `if` statements can be nested to any depth.

```
if (condition_1) {  
    statements  
    if (condition_2) {  
        statements  
    }  
}
```

- Tip: (Too) Deep nesting is bad practice. If you have deep nesting, you can try to redesign the tests performed in the if clauses.

- *nested\_if.c*

```

1#include <stdio.h>
2// nested_if.c
3int main () {
4
5    /* local variable definition */
6    int a;
7    int b;
8
9    printf("Enter an integer number (<= 100)");
10   scanf("%d", &a);
11
12   /* check the boolean condition */
13   if( a == 100 ) {
14
15       /* if condition is true then read another integer */
16
17       printf("Enter another integer number (<= 200)");
18       scanf("%d", &b);
19
20       if( b == 200 ) {
21           /* if condition is true then print the following */
22           printf("Value of a is 100 and b is 200\n" );
23       }
24   }
25
26   printf("Exact value of a is : %d\n", a );
27   printf("Exact value of b is : %d\n", b );
28
29   return 0;
30}

```

b not initialized!

```

frankvp@CRD-L-08004:~/Controlflow$ gcc nested_if.c -o nested_if
frankvp@CRD-L-08004:~/Controlflow$ ./nested_if
Enter an integer number (<= 100)56
Exact value of a is : 56
Exact value of b is : 32765
frankvp@CRD-L-08004:~/Controlflow$ ./nested_if
Enter an integer number (<= 100)23
Exact value of a is : 23
Exact value of b is : 32765
frankvp@CRD-L-08004:~/Controlflow$ ./nested_if
Enter an integer number (<= 100)-99
Exact value of a is : -99
Exact value of b is : 32764
frankvp@CRD-L-08004:~/Controlflow$

```

KU LEUVEN

16

## if - else construct

- has the form

```

if (condition){
    statement(s)_true;
}
else {
    statement(s)_false;
}

```

- If *<condition>* evaluates to TRUE, then *<statement(s)\_true>* is executed. If FALSE, then *<statement(s)\_false>* is executed.

KU LEUVEN

17

### demo\_ifelse.c

```
1 /* demo_ifelse.c
2 */
3 #include <stdio.h>
4
5 void main()
6 {
7     int number=0;
8
9     printf("Enter number in between -20 and 20 ");
10
11     scanf("%d", &number);
12
13     if (number > 0)
14         printf("%d is a positive number \n", number);
15     else {
16         printf("%d is a negative number \n", number);
17         printf("positive numbers are better \n");
18     }
19     printf("the end \n");
20
21 }
```

```
frankvp@CRD-L-08004:~/Controlflow$ gcc demo_ifelse.c -o demo_ifelse
frankvp@CRD-L-08004:~/Controlflow$ ./demo_ifelse
Enter number in between -20 and 20 0
0 is a negative number
positive numbers are better
the end
frankvp@CRD-L-08004:~/Controlflow$ ./demo_ifelse
Enter number in between -20 and 20 5
5 is a positive number
the end
frankvp@CRD-L-08004:~/Controlflow$ ./demo_ifelse
Enter number in between -20 and 20 -9
-9 is a negative number
positive numbers are better
the end
frankvp@CRD-L-08004:~/Controlflow$
```

KU LEUVEN

18

### high\_low.c

```
/* high_low.c */
/* based on Todd RPI */

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

void main()
{
    int number, choice;

    srand(time(NULL));
    number = 1 + rand() % 10;

    printf("Choose number between 1 and 10: ");
    scanf("%d", &choice);

    if (choice < number)
        printf("too low! \n");
    else if (choice > number)
        printf("too high! \n");
    else
        printf("well done! \n");

    printf("the correct number is %d \n", number);
}
```

```
frankvp@CRD-L-08004:~/Controlflow$ gcc high_low.c -o high_low
frankvp@CRD-L-08004:~/Controlflow$ ./high_low
Choose number between 1 and 10: 2
too low!
the correct number is 9
frankvp@CRD-L-08004:~/Controlflow$ ./high_low
Choose number between 1 and 10: 3
too high!
the correct number is 1
frankvp@CRD-L-08004:~/Controlflow$ ./high_low
Choose number between 1 and 10: 3
too low!
the correct number is 8
frankvp@CRD-L-08004:~/Controlflow$ ./high_low
Choose number between 1 and 10: 3
too low!
the correct number is 7
frankvp@CRD-L-08004:~/Controlflow$ ./high_low
Choose number between 1 and 10: 3
too low!
the correct number is 5
frankvp@CRD-L-08004:~/Controlflow$
```

19

## Dangling else

- Anything wrong here?

```
if (n >= 0)
    if (a > n)
        n = a;
else
    n = 0;
```

- The **else** associates with the innermost (next) **if**
- Use braces **{ }** to force association:

```
if (n >= 0){
    if (a > n)
        n = a;
}
else
    n = 0;
```

## Conditional Expression ? :

- A ternary operator as it takes 3 operands

$(expr\ 0) ? (expr\ 1) : (expr\ 2)$

- Evaluate *expr 0*, and if TRUE: evaluate *expr 1*, else evaluate *expr 2*
- The result of the ternary expression is either *expr 1* or *expr 2*
- Behaves like an **if-else** in a single expression

```
c = (a < b) ? a : b; /* c = min(a,b) */
x = (x < 0) ? -x : x; /* x = abs(x) */
```

- File: *conditionalexpr.c*



# else-if construct

- has the form

```
if (expression_0){
    statement_0;}
else if (expression_1){
    statement_1;}
else if (expression_2){
    statement_2;}
else {
    statement_n;}
```

A general selection structure for a multi-way decision

- Can be any number of **else ifs**
- Final **else** is a default if all conditionals above are FALSE.
- Final **else** is optional, but good practice to use it for “can’t happen” situations – to catch bugs.

```
1/*
2 calc1.c
3
4 This program implements a *very* simple 4-function calculator that
5 evaluates expressions of the form:
6     <number> <operator> <number>
7 The allowable operators are (+, -, *, /)
8 */
9 #include <stdio.h>
10
11 int main(void)
12 {
13     double val1, val2, result;
14     unsigned char op;
15     int isok = 1;
16
17     printf("Enter your expression.\n");
18     scanf ("%lf %c %lf", &val1, &op, &val2);
19
20     if (op == '+')
21         result = val1 + val2;
22     else if (op == '-')
23         result = val1 - val2;
24     else if (op == '*')
25         result = val1 * val2;
26     else if (op == '/') {
27         if (val2 == 0) {
28             printf("ERROR. Divide by zero.\n");
29             isok = 0;
30         }
31         else
32             result = val1 / val2;
33     }
34     else {
35         printf("ERROR. Invalid operator.\n");
36         isok = 0;
37     }
38
39     if (isok)
40         printf("%f\n", result);
41     return 0;
42 }
```

## • calc1.c

```
frankvp@CRD-L-08004:~/Controlflow$ gcc calc1.c -o calc1
frankvp@CRD-L-08004:~/Controlflow$ ./calc1
Enter your expression.
5 * 3.12
15.600000
frankvp@CRD-L-08004:~/Controlflow$ ./calc1
Enter your expression.
3 - 2
1.000000
frankvp@CRD-L-08004:~/Controlflow$ ./calc1
Enter your expression.
3 ** 5
0.000000
frankvp@CRD-L-08004:~/Controlflow$ ./calc1
Enter your expression.
3 z 5
ERROR. Invalid operator.
frankvp@CRD-L-08004:~/Controlflow$
```

what happened?

- Check in debugger
- \*\* is parsed as \*

```

calc1.c
12      {
13          double val1, val2, result;
14          unsigned char op;
15          int isok = 1;
16
17          printf("Enter your expression.\n");
18          scanf ("%lf %c %lf", &val1, &op, &val2);
19
20      if (op == '+')
21          result = val1 + val2;
22      else if (op == '-')
23          result = val1 - val2;
24      else if (op == '*')
25          result = val1 * val2;
26      else if (op == '/') {
27          if (val2 == 0) {
28              printf("ERROR. Divide by zero.\n");
29              isok = 0;
30          }
31      } else

```

native process 2280 In: main  
(gdb) b 20  
Breakpoint 1 at 0x11f7: file calc1.c, line 20.  
(gdb) run  
Starting program: /mnt/c/temp/Develop/CDev/Controlflow/calc1  
Breakpoint 1, main () at calc1.c:20  
(gdb) p op  
\$1 = 42 '\*'  
(gdb) p val2  
\$2 = 4.6355705384986992e-310  
(gdb)

24

## switch statement

- format

```

switch(integer expression)
{
    case <int_0>:
        statements
        break;
    case <int_1>:
        statements
        break;
    default:
        statements
        break;
}

```

- Expression must evaluate to an integer (char, int)
- no {} blocks within each case.
- the colon : for each case and value.
- The “condition” of a switch statement is a value.
- The default case is optional, but useful to handle unexpected cases.
- Do not forget the break statement. Otherwise fall through to the next case

25

```

1#include <stdio.h>
2// calc2.c
3int main(void)
4{
5    double val1, val2, result;
6    unsigned char op;
7    int isok = 1;
8
9    printf("Enter your expression.\n");
10   scanf ("%lf %c %lf", &val1, &op, &val2);
11
12   switch (op)
13   {
14       case '+':
15           result = val1 + val2;
16           break;
17
18       case '-':
19           result = val1 - val2;
20           break;
21
22       case '*':
23           result = val1 * val2;
24           break;
25
26       case '/':
27           if (val2 == 0) {
28               printf("ERROR. Divide by zero.\n");
29               isok = 0;
30           }
31           else
32               result = val1 / val2;
33           break;
34
35       default:
36           printf("ERROR. Invalid operator.\n");
37           isok = 0;
38           break;
39   }
40
41   if (isok)
42       printf("%f\n", result);
43
44   return 0;
45}

```

## • calc2.c

```

frankvp@CRD-L-08004:../Controlflow$ gcc calc2.c -o calc2
frankvp@CRD-L-08004:../Controlflow$ ./calc2
Enter your expression.
3 * 7.3
21.900000
frankvp@CRD-L-08004:../Controlflow$ ./calc2
Enter your expression.
3.3 - 7.4
-4.100000
frankvp@CRD-L-08004:../Controlflow$ ./calc2
Enter your expression.
3 ** 2
0.000000
frankvp@CRD-L-08004:../Controlflow$

```

problem!

ICTS

KU LEUVEN

26

```

1/* calculator */
2/* based on Todd RPI */
3
4#include<stdio.h>
5
6int main()
7{
8    float oper1, oper2;
9    int operator;
10    float result;
11
12    printf("enter Number1: ");
13    scanf("%g", &oper1);
14    printf("enter Number2: ");
15    scanf("%g", &oper2);
16
17    printf(" enter the operation (1: +, 2: -, 3: *)");
18    scanf("%d", &operator);
19
20    switch (operator)
21    {
22        case 1:
23            result = oper1 + oper2;
24            break;
25        /*
26        */
27        /* carfeul with break */
28
29        case 2:
30            result = oper1 - oper2;
31            break;
32
33        case 3:
34            result = oper1 * oper2;
35            break;
36
37        default:
38            printf("illegal operator \n");
39            return(1);
40            break;
41    }
42    printf (" result is %g \n", result);
43}

```

```

frankvp@CRD-L-08004:../Controlflow$ gcc calculator.c -o calculator
frankvp@CRD-L-08004:../Controlflow$ ./calculator
enter Number1: 2
enter Number2: 3.6
enter the operation (1: +, 2: -, 3: *)3
result is 7.2
frankvp@CRD-L-08004:../Controlflow$ ./calculator
enter Number1: 2.2
enter Number2: 3.78
enter the operation (1: +, 2: -, 3: *)7
illegal operator
frankvp@CRD-L-08004:../Controlflow$

```

ICTS

KU LEUVEN

27

# switch statement

- limitation:

- no testing on strings

`switch("Test")    switch(strName)`

`switch()` statements expect constant values

- `switch()` statements do not work with floating point numbers

`switch(22.2)`

- no expressions in case labels

`case (number != 2)`

- <https://data-flair.training/blogs/limitation-and-disadvantages-of-switch-case-in-c/>