KU LEUVEN

C: an introduction

Control flow: Conditionals

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

control flow

- Two main aspects of a C 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);}`

KU LEUVEN

`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 {})
- Indentation not needed for compiler
  - Needed for readability

KU LEUVEN

What are True and False?

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

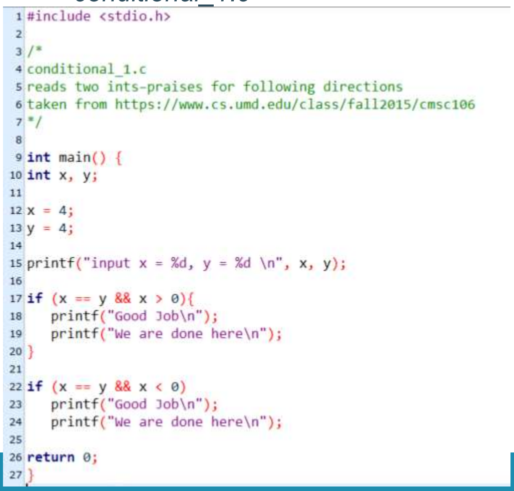
  ```
  c = (a <= b);
  ```

- 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.)

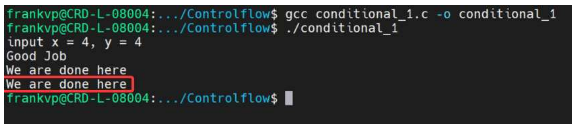
  ```
  if (b - a) { … }
  ```

KU LEUVEN

conditional_1.c

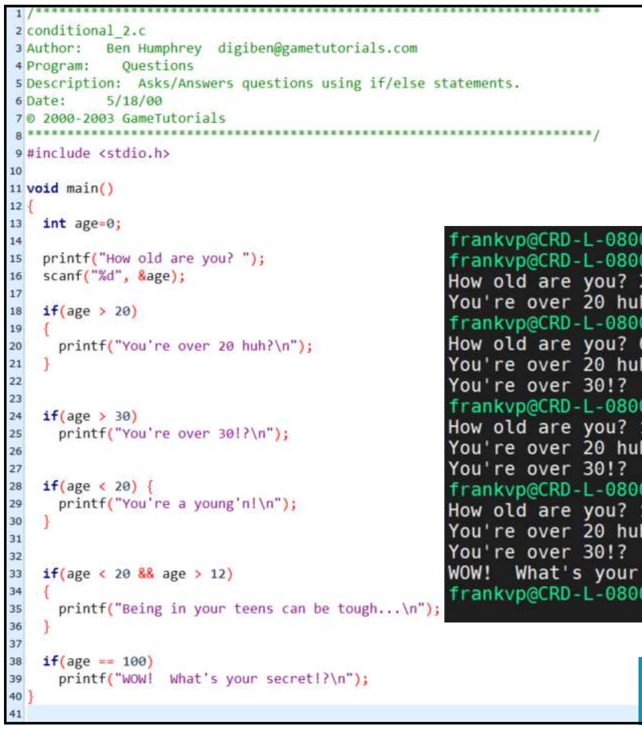


```c
#include <stdio.h>

/*
conditional_1.c
reads two ints-praises for following directions
taken from https://www.cs.umd.edu/class/fall2015/cmsc106
*/

int main() {
int x, y;

x = 4;
y = 4;

printf("input x = %d, y = %d \n", x, y);

if (x == y && x > 0){
   printf("Good Job\n");
   printf("We are done here\n");
}

if (x == y && x < 0)
   printf("Good Job\n");
   printf("We are done here\n");

return 0;
}
```



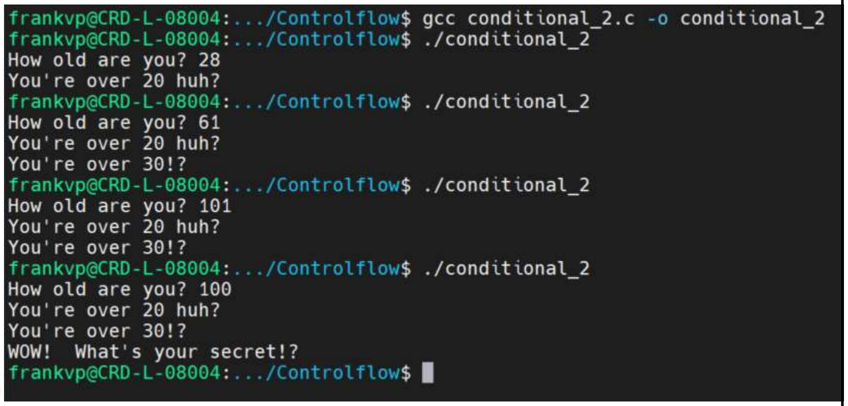
```
frankvp@CRD-L-08004:.../Controlflow$ gcc conditional_1.c -o conditional_1
frankvp@CRD-L-08004:.../Controlflow$ ./conditional_1
input x = 4, y = 4
Good Job
We are done here
We are done here
frankvp@CRD-L-08004:.../Controlflow$
```

KU LEUVEN



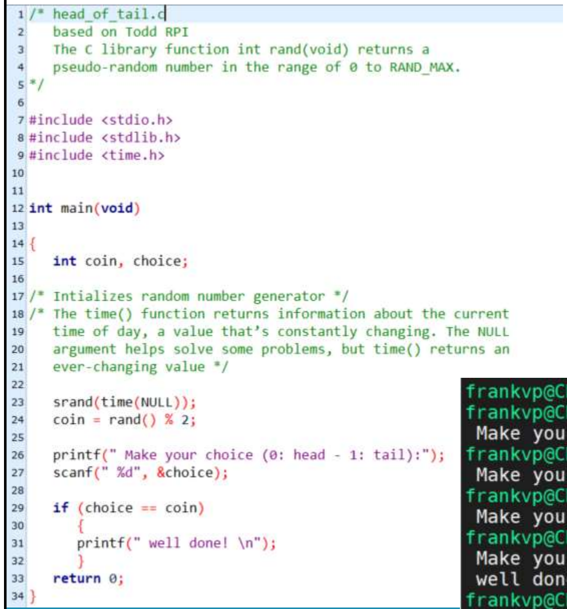
```c
/****************************************************************************
conditional_2.c
Author:   Ben Humphrey  digiben@gametutorials.com
Program:    Questions
Description:  Asks/Answers questions using if/else statements.
Date:     5/18/00
© 2000-2003 GameTutorials
****************************************************************************/
#include <stdio.h>

void main()
{
  int age=0;

  printf("How old are you? ");
  scanf("%d", &age);

  if(age > 20)
  {
    printf("You're over 20 huh?\n");
  }


  if(age > 30)
    printf("You're over 30!?\n");


  if(age < 20) {
    printf("You're a young'n!\n");
  }


  if(age < 20 && age > 12)
  {
    printf("Being in your teens can be tough...\n");
  }

  if(age == 100)
    printf("WOW!  What's your secret!?\n");
}
```

• *conditional_2.c*



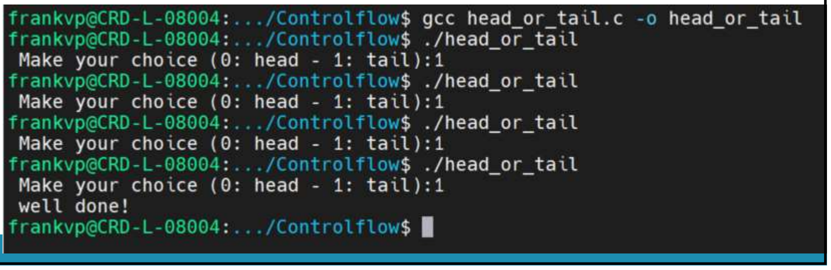
```
frankvp@CRD-L-08004:.../Controlflow$ gcc conditional_2.c -o conditional_2
frankvp@CRD-L-08004:.../Controlflow$ ./conditional_2
How old are you? 28
You're over 20 huh?
frankvp@CRD-L-08004:.../Controlflow$ ./conditional_2
How old are you? 61
You're over 20 huh?
You're over 30!?
frankvp@CRD-L-08004:.../Controlflow$ ./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



```c
/* head_of_tail.c
   based on Todd RPI
   The C library function int rand(void) returns a
   pseudo-random number in the range of 0 to RAND_MAX.
*/

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


int main(void)

{
   int coin, choice;

/* Intializes random number generator */
/* The time() function returns information about the current
   time of day, a value that's constantly changing. The NULL
   argument helps solve some problems, but time() returns an
   ever-changing value */

   srand(time(NULL));
   coin = rand() % 2;

   printf(" Make your choice (0: head - 1: tail):");
   scanf(" %d", &choice);

   if (choice == coin)
      {
      printf(" well done! \n");
      }
   return 0;
}
```

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$
```

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 `{ }`

KU LEUVEN

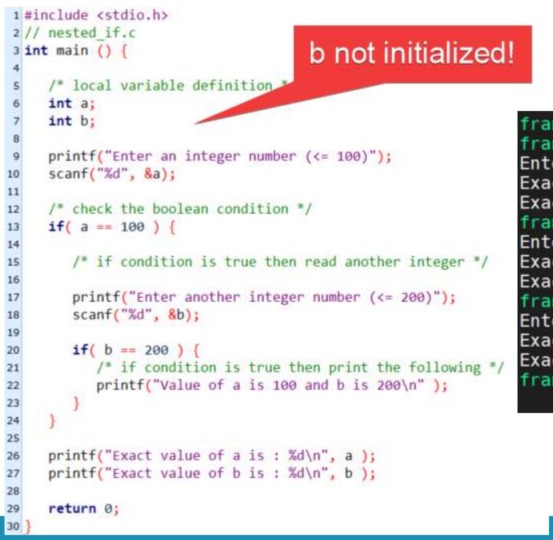
Nested `if` statements

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

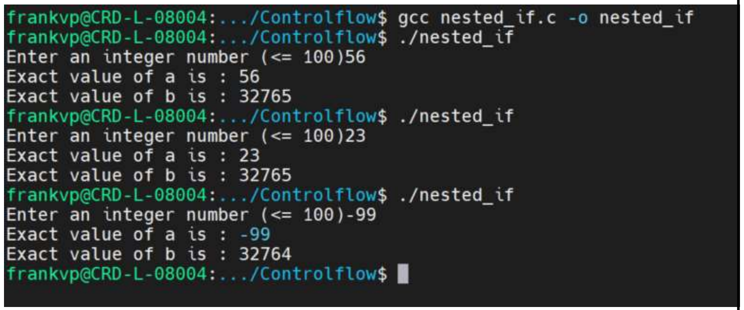
```
if (condition_1) {
    statements
    if (condition_2) {
        statements
    }
}
```
- (Too) Deep nesting is bad practice. If you have deep nesting, you can try to redesign the tests performed in the if clauses,

KU LEUVEN

- *nested_if.c*



```c
#include <stdio.h>
// nested_if.c
int main () {

   /* local variable definition */
   int a;
   int b;

   printf("Enter an integer number (<= 100)");
   scanf("%d", &a);

   /* check the boolean condition */
   if( a == 100 ) {

      /* if condition is true then read another integer */

      printf("Enter another integer number (<= 200)");
      scanf("%d", &b);

      if( b == 200 ) {
         /* if condition is true then print the following */
         printf("Value of a is 100 and b is 200\n" );
      }
   }

   printf("Exact value of a is : %d\n", a );
   printf("Exact value of b is : %d\n", b );

   return 0;
}
```



```
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

`if - else` construct

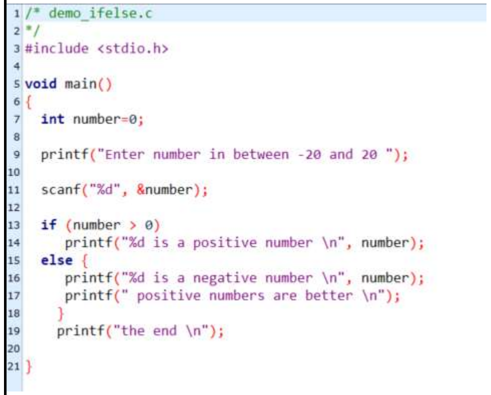
- has the form

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

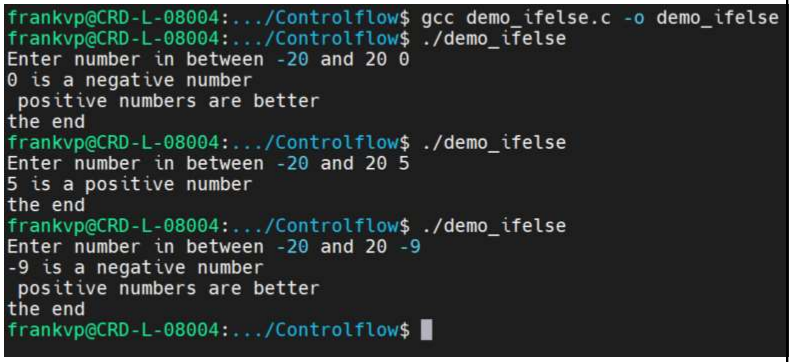
- If `<condition>` evaluates to TRUE, then `<statement(s)_true>` is executed. If FALSE, then `<statement(s)_false>` is executed.

KU LEUVEN

demo_ifelse.c



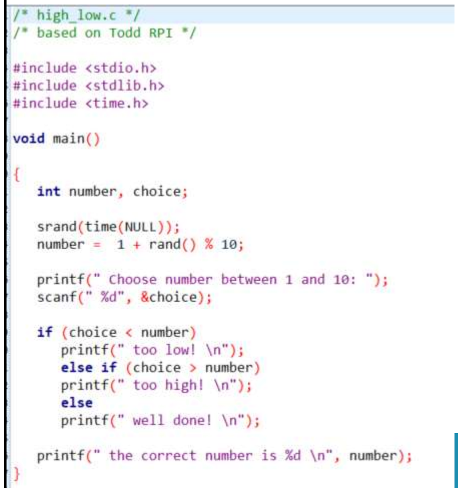
```c
/* demo_ifelse.c
*/
#include <stdio.h>

void main()
{
  int number=0;

  printf("Enter number in between -20 and 20 ");

  scanf("%d", &number);

  if (number > 0)
     printf("%d is a positive number \n", number);
  else {
     printf("%d is a negative number \n", number);
     printf(" positive numbers are better \n");
    }
    printf("the end \n");

}
```



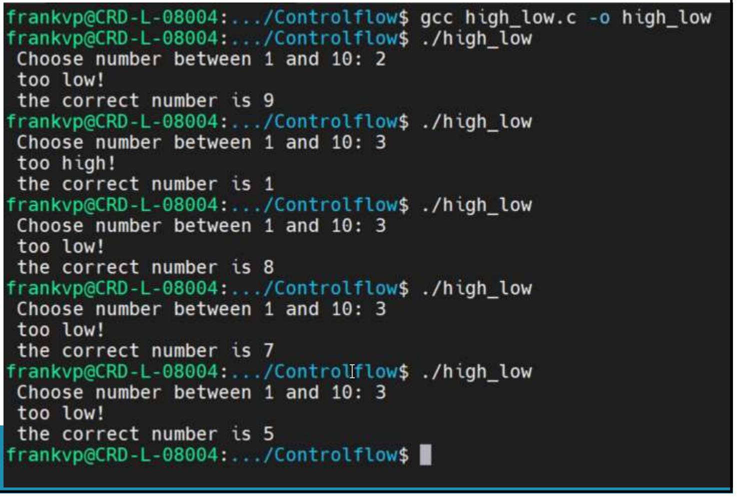
```
frankvp@CRD-L-08004:.../Controlflow$ gcc demo_ifelse.c -o demo_ifelse
frankvp@CRD-L-08004:.../Controlflow$ ./demo_ifelse
Enter number in between -20 and 20 0
0 is a negative number
 positive numbers are better
the end
frankvp@CRD-L-08004:.../Controlflow$ ./demo_ifelse
Enter number in between -20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

high_low.c



```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$
```

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;
```

KU LEUVEN

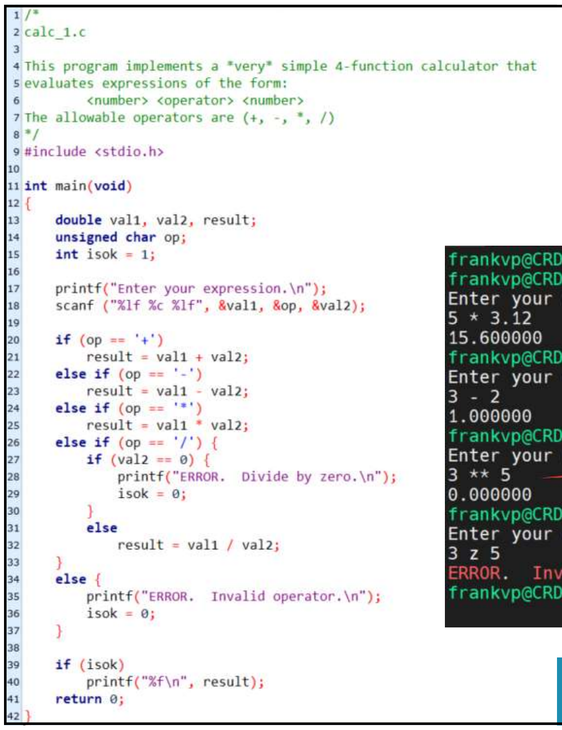
else-if construct

- has the form

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

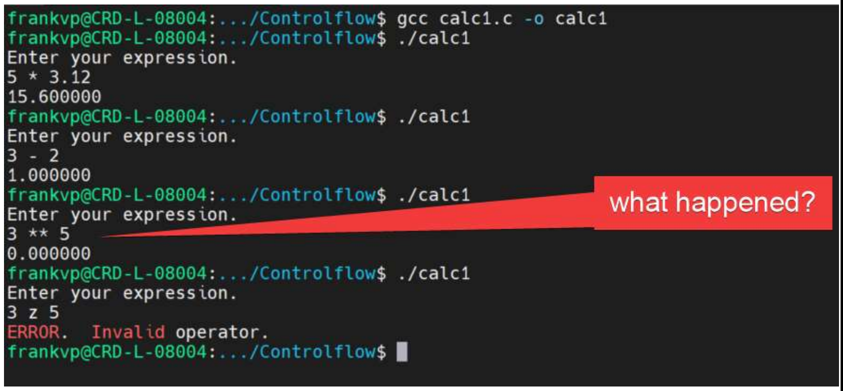
A general selection structure for a multi-way decision
- Can be any number of **else ifs**
- Final **else** is a default if all conditionals above are FALSE.
- Final **else** is optional, but good practice to use it for “can’t happen” situations – to catch bugs.

KU LEUVEN



```c
/*
calc_1.c

This program implements a *very* simple 4-function calculator that
evaluates expressions of the form:
        <number> <operator> <number>
The allowable operators are (+, -, *, /)
*/
#include <stdio.h>

int main(void)
{
    double val1, val2, result;
    unsigned char op;
    int isok = 1;

    printf("Enter your expression.\n");
    scanf ("%lf %c %lf", &val1, &op, &val2);

    if (op == '+')
        result = val1 + val2;
    else if (op == '-')
        result = val1 - val2;
    else if (op == '*')
        result = val1 * val2;
    else if (op == '/') {
        if (val2 == 0) {
            printf("ERROR.  Divide by zero.\n");
            isok = 0;
        }
        else
            result = val1 / val2;
    }
    else {
        printf("ERROR.  Invalid operator.\n");
        isok = 0;
    }

    if (isok)
        printf("%f\n", result);
    return 0;
}
```

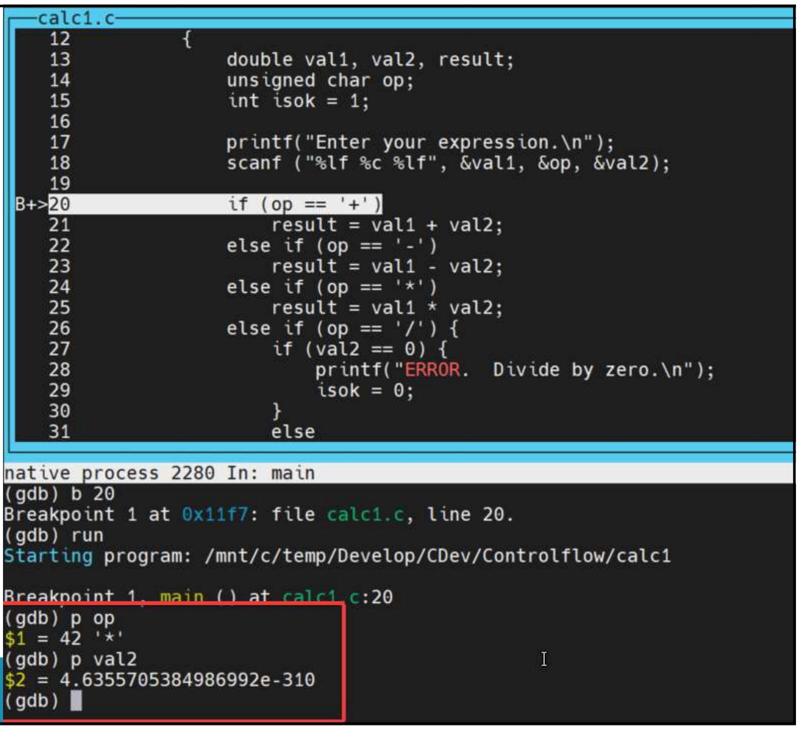
- **calc1.c**



```
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$
```

KU LEUVEN

- 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
B+> 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)
```

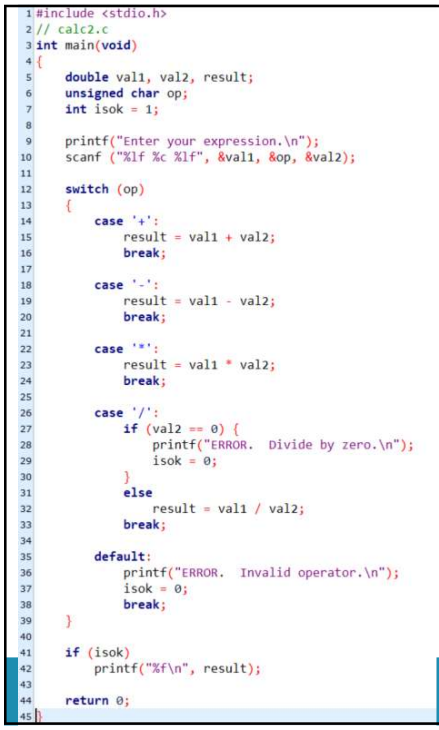
`switch` statement

- format

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

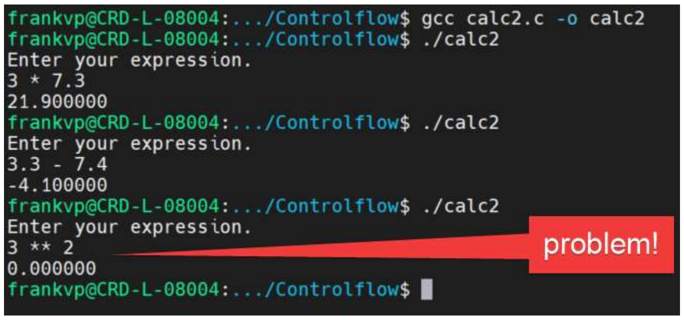
- no {} blocks within each case.
- the colon : for each case and value.
- The “condition” of a switch statement is a value.
- The `default` case is optional, but useful to handle unexpected cases.
- Do not forget the `break` statement.
  Otherwise fall through to the next case

KU LEUVEN



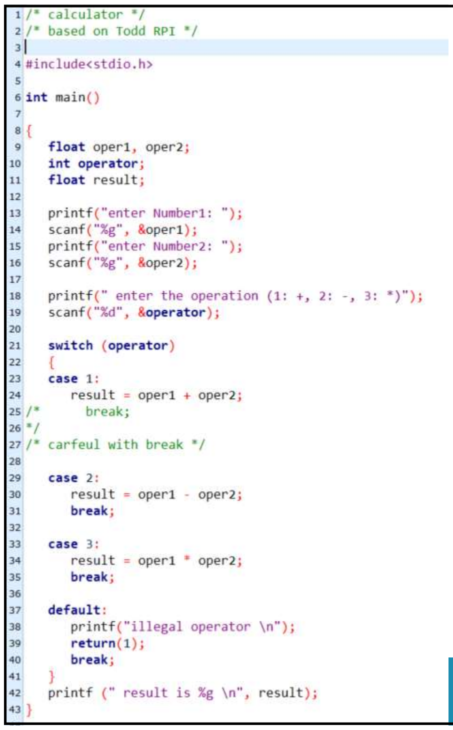
```c
#include <stdio.h>
// calc2.c
int main(void)
{
    double val1, val2, result;
    unsigned char op;
    int isok = 1;

    printf("Enter your expression.\n");
    scanf ("%lf %c %lf", &val1, &op, &val2);

    switch (op)
    {
        case '+':
            result = val1 + val2;
            break;

        case '-':
            result = val1 - val2;
            break;

        case '*':
            result = val1 * val2;
            break;

        case '/':
            if (val2 == 0) {
                printf("ERROR.  Divide by zero.\n");
                isok = 0;
            }
            else
                result = val1 / val2;
            break;

        default:
            printf("ERROR.  Invalid operator.\n");
            isok = 0;
            break;
    }

    if (isok)
        printf("%f\n", result);

    return 0;
}
```

- *calc2.c*

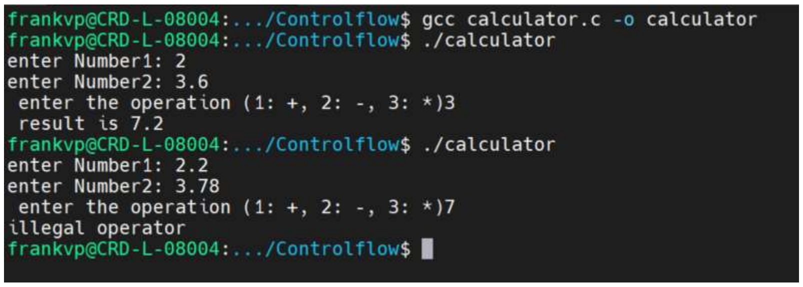


```
frankvp@CRD-L-08004:.../Controlflow$ gcc calc2.c -o calc2
frankvp@CRD-L-08004:.../Controlflow$ ./calc2
Enter your expression.
3 * 7.3
21.900000
frankvp@CRD-L-08004:.../Controlflow$ ./calc2
Enter your expression.
3.3 - 7.4
-4.100000
frankvp@CRD-L-08004:.../Controlflow$ ./calc2
Enter your expression.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

ICTS KU LEUVEN



```c
/* calculator */
/* based on Todd RPI */

#include<stdio.h>

int main()

{
   float oper1, oper2;
   int operator;
   float result;

   printf("enter Number1: ");
   scanf("%g", &oper1);
   printf("enter Number2: ");
   scanf("%g", &oper2);

   printf(" enter the operation (1: +, 2: -, 3: *)");
   scanf("%d", &operator);

   switch (operator)
   {
   case 1:
      result = oper1 + oper2;
/*       break;
*/
/* carfeul with break */

   case 2:
      result = oper1 - oper2;
      break;

   case 3:
      result = oper1 * oper2;
      break;

   default:
      printf("illegal operator \n");
      return(1);
      break;
   }
   printf (" result is %g \n", result);
}
```



```
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

`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)`

KU LEUVEN