CSE1142 – Selective Structures in C

Sanem Arslan Yılmaz

Some of the slides are from:
CMPE150 – Boğaziçi University
Deitel & Associates

Agenda

- Algorithms & Pseudocode
- Flowcharts
- If statement
- Else-if statement
- Ternary operator
- Switch statement
- Break statement

Introduction

- Before writing a program to solve a particular problem, we must have a thorough understanding of the problem and a carefully planned solution approach.

Algorithms

- The solution to any computing problem involves executing a series of actions in a specific order.
- A procedure for solving a problem in terms of
  - the actions to be executed, and
  - the order in which these actions are to be executed

  is called an algorithm.
- Correctly specifying the order in which the actions are to be executed is important.

Algorithms (Cont.)

- Consider the "rise-and-shine algorithm" followed by one junior executive for getting out of bed and going to work:
  - (1) Get out of bed,
  - (2) take off pajamas,
  - (3) take a shower,
  - (4) get dressed,
  - (5) eat breakfast,
  - (6) carpool to work.
- This routine gets the executive to work well prepared to make critical decisions.
- Specifying the order in which statements are to be executed in a computer program is called program control.

Pseudocode

- Pseudocode is an artificial and informal language that helps you develop algorithms.
- Pseudocode is similar to everyday English; it's convenient and user friendly although it's not an actual computer programming language.
- Pseudocode programs are *not* executed on computers.
- Rather, they merely help you "think out" a program before attempting to write it in a programming language like C.

Pseudocode (Cont.)

- A carefully prepared pseudocode program may be converted easily to a corresponding C program.
- Pseudocode consists only of action and decision statements—those that are executed when the program has been converted from pseudocode to C and is run in C.
- Definitions are not executable statements.
- They’re simply messages to the compiler.

Flowcharts

- A flowchart is a graphical representation of an algorithm or of a portion of an algorithm.
- Like pseudocode, flowcharts are useful for developing and representing algorithms, although pseudocode is preferred by most programmers.
- Flowcharts are drawn using certain special-purpose symbols such as rectangles, diamonds, rounded rectangles, and small circles; these symbols are connected by arrows called flowlines.

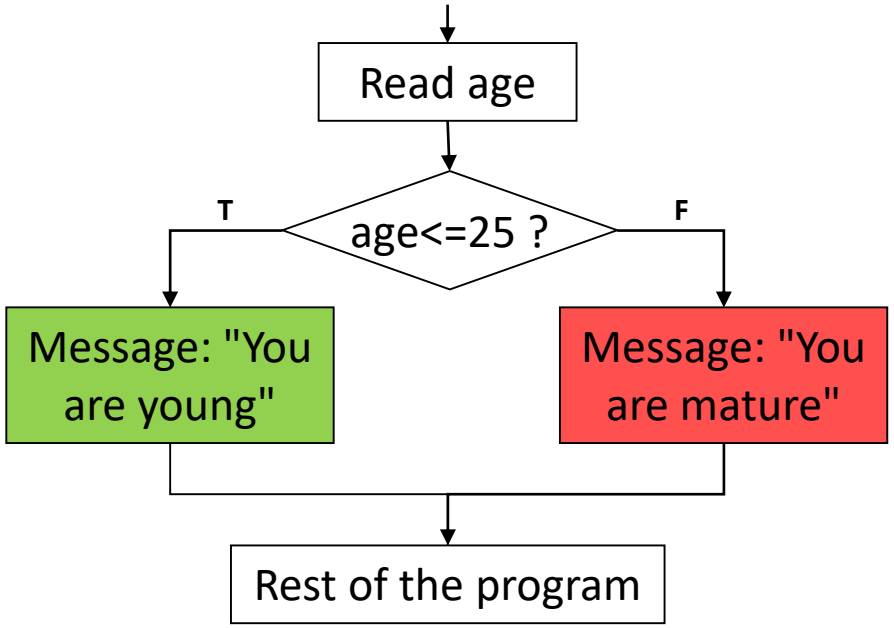
Flowcharts

- We use the rectangle symbol, also called the action symbol, to indicate any type of action including a calculation or an input/output operation.
- Perhaps the most important flowcharting symbol is the diamond symbol, also called the decision symbol, which indicates that a decision is to be made.
- The flowlines indicate the order in which the actions are performed.

If statement

- The "if statement" is used to break the sequential flow of execution.

- Enforces branching in execution according to the result of an expression.
  - There are two possible paths to take.
  - The expression determines which path should be taken.



If statement syntax

- Syntax:

```
if (int_expr)
    stat_block1
else
    stat_block2
```

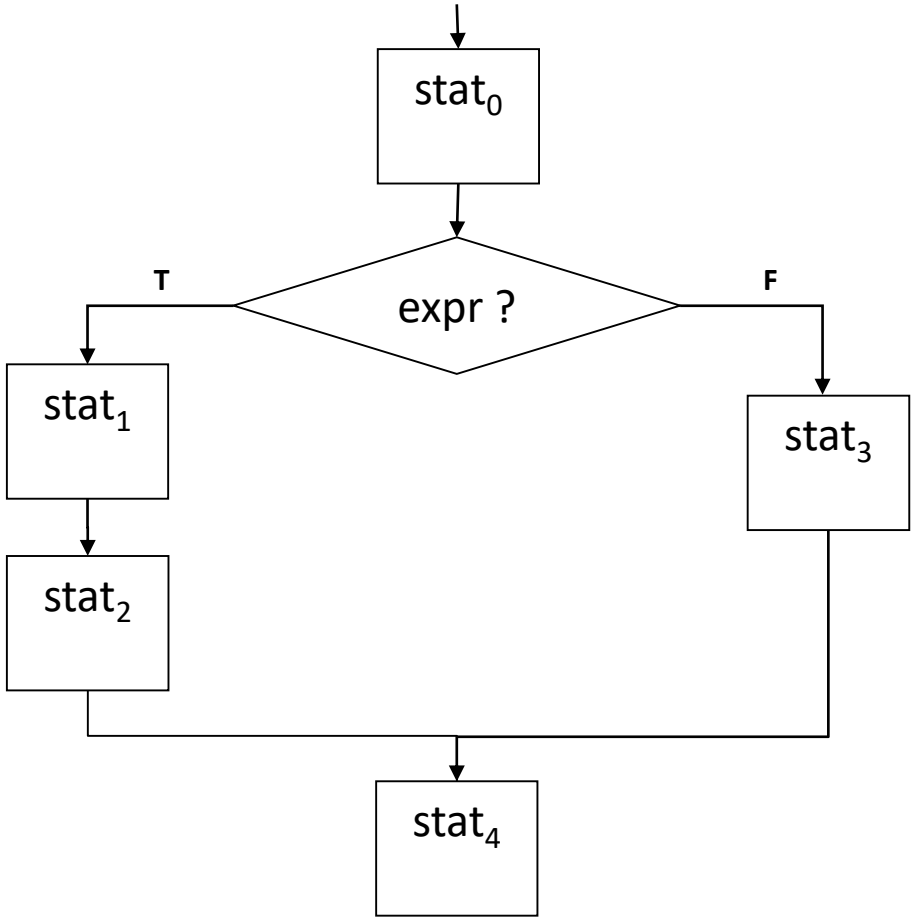
where **stat_block** is one of the following:

- a single statement `stat;`
- the null statement `;`
- a group of statements enclosed in braces

```
{
    stat1;
    ...
    statn;
}
```

If statement example I

```
stat0;
if (expr)
{   stat1;
    stat2;
}
else
    stat3;
stat4;
```



Notice the indentation.

Coding style

- Note that if and else blocks are inside the main block, so they are *indented* further.
- Make sure ***if*** and ***else*** keywords are aligned.
- Statements inside parallel blocks are aligned.
- These are conventions, not syntax rules. C compiler disregards any indentation

```
#include <stdio.h>
int main()
{
        int grade;
        printf("Enter grade: ");
        scanf("%d", &grade);

        if (grade > 50) {
                printf("Passed.\n");
        }
        else {
                printf("Failed.\n");
        }
}
```

Blocks with a single statement

- When an *if* or *else* block has a single statement, braces can be omitted.

```
if (grade > 50) {
        printf("Passed.\n");
}
else {
        printf("Failed.\n");
}
```

same as

```
if (grade > 50)
        printf("Passed.\n");
else
        printf("Failed.\n");
```

Do not omit braces in multiline blocks

- Suppose we want to count the students that passed.

```
if (grade > 50){
        printf("Passed.\n");
        number_passed += 1;
}
```

- If we omit the braces this becomes

```
if (grade > 50)
        printf("Passed.\n");
        number_passed += 1;
```

- Equivalent to

```
if (grade > 50){
        printf("Passed.\n");
}
number_passed ++;
```

- The number is increased for any grade.

No semicolon after if or else

- A common mistake is to put a semicolon at the end of the ***if*** line:

```
if (grade > 50);
        printf("Passed.\n");
```

- This is valid code; no syntax error. It is the same as:

```
if (grade > 50)
{
        /* empty block */
}
printf("Passed.\n");
```

- Thus, "Passed" is printed for any grade. Not what we want!

If statement example II

- Read a number and state whether it is odd or even.

```
int num;

scanf("%d", &num);
printf("%d is an ", num);
if (num % 2 != 0)
   printf("odd ");
else
   printf("even ");
printf("number.\n");
```

If statement example III

- Read one character as input; check if it is a digit; if so, convert it to an integer and display the number that is two times the input number; o/w display an error message.

```
char ch;   int num;

scanf("%c", &ch);
if (('0' <= ch) && (ch <= '9')){
   num = ch - '0';
   printf("Two times the input is %d \n", 2 * num);
}
else
   printf("Input is not a digit! \n");
```

Else-if statement

```
if (age <= 1)
   printf("infant");
else if (age <= 3)
   printf("toddler");
else if (age <= 10)
   printf("child");
else if (age <= 18)
   printf("adolescent");
else if (age <= 25)
   printf("young");
else if (age <= 39)
   printf("adult");
else if (age <= 65)
   printf("middle-aged");
else
   printf("elderly");
```

Ternary operator

- Ternary operator is similar to the "if" statement. But it is an operator, not a statement.

- Syntax:

```
<condition> ? <expr_1> : <expr_2>
```

- Eg:

```
a = (b > c) ? b : c;
k = (n != 0) ? m/n : 0;
```

Switch statement

- If you have multiple cases depending on different values of the same integer expression, switch is easier to use.

- Syntax:

```
switch (int_expr)
{  case constant_int_value1: stat(s);
   case constant_int_value2: stat(s);
   ...
   default: stat(s);
}
```

- You may have zero or more statements in each case.

Break statement

- Switch statement actually gathers many statements of several cases.
  - The case labels denote the specific statement from which the execution of this group of statements begins.
  - All statements till the end of the group are executed sequentially.
- To separate the cases, **`break`** statement is used.
  - **`break`** breaks the sequential execution of the statements and immediately jumps to the end of the switch statement.

Switch statement example

- Define the days of the week as an enumerated type and display them as strings.

```
enum day_type {MON=1,TUE,WED,THU,FRI,SAT,SUN} day;

scanf("%d", &day);
switch (day)
{  case SUN: printf("Sunday\n"); break;
   case WED: printf("Wednesday\n"); break;
   case TUE: printf("Tuesday\n"); break;
   case THU: printf("Thursday\n"); break;
   case FRI: printf("Friday\n"); break;
   case SAT: printf("Saturday\n"); break;
   case MON: printf("Monday\n"); break;
   default:  printf("Incorrect day\n"); break;
}
```