Selection in C++

CS1A

- Flowcharts to code
- IF-THEN
  - NESTED IF-THEN statements
- IF-THEN-ELSE
  - NESTED IF-THEM-ELSE

3 Basic Control / Logic Structures

- Sequence
  - Instructions are executed one after another in the order they appear in the program
  - Until another control structure takes precedence
- Selection
  - Based on some condition, either one part of the program is executed or another part is executed
  - The program chooses which part to execute based on the condition
- Repetition
  - Part of the code is executed over and over (repeated)
  - This can be for a set number of times or until a condition is met

This is what we have been doing thus far

Today we will focus on Selection.
Don't worry → We'll get to the others later on.

Selection Structures

What if I only want some instructions to run some of the time?

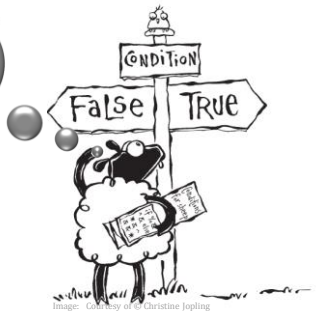


Image: Courtesy of © Christine Jopling

Selection Structures

Selection
→ Choosing between two or more alternative actions

- Run certain instructions based on some condition
- Conditions are based on Boolean Expressions
  - An expression that evaluates to 1 of 2 possibilities
  - Either True or False
- The computer evaluates a Boolean Expression and determines which instructions to execute based on the result
- Boolean expressions are formed using relational operators

Relational Operators

We use Relational Operators to form Boolean Expressions

NOTE: this is not the same as =, = ← is an assignment

==	Equal
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal
!=	Not Equal

We use Relational Operators to compare values in Selection Statements
→These will return a True (1) or False (0) value.

Examples of Boolean Expressions

What will be the result of these Boolean functions?

5 == 5
'a' < 'c'
4 + 3 > 10
10 != 20
6 <= 6
5 >= 9
'A' < 'Z'
'a' < 'Z'

If you compare characters using relational operators,
→It compares the ASCII values
(in this case 'a' has a greater ASCII value than 'Z'

ASCII Chart for Printing Characters

Char	Decimal Value	Char	Dec	Char	Dec	Char	Dec
SP	32	8	56	P	80	h	104
!	33	9	57	Q	81	i	105
"	34	:	58	R	82	j	106
#	35	;	59	S	83	k	107
$	36	<	60	T	84	l	108
%	37	=	61	U	85	m	109
&	38	>	62	V	86	n	110
`	39	?	63	W	87	o	111
(	40	@	64	X	88	p	112
)	41	A	65	Y	89	q	113
*	42	B	66	Z	90	r	114
+	43	C	67	[	91	s	115
,	44	D	68	\	92	t	116
-	45	E	69	]	93	u	117
.	46	F	70	^	94	v	118
/	47	G	71	_	95	w	119
0	48	H	72	`	96	x	120
1	49	I	73	a	97	y	121
2	50	J	74	b	98	z	122
3	51	K	75	c	99	{	123
4	52	L	76	d	100	|	124
5	53	M	77	e	101	}	125
6	54	N	78	f	102	~	126
7	55	O	79	g	103	DEL	127

3-types of Selection Statements

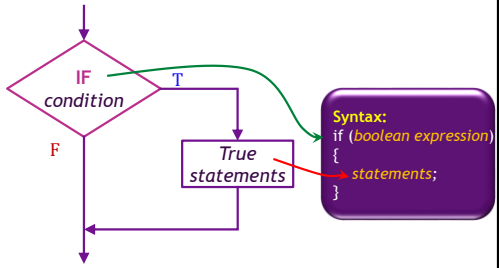
- One-way Decisions
  - If-Then Statements
  - If the condition is true then execute some instructions
  - If the condition is false → don't do anything special
- Two-way Decisions
  - If-Then-Else Statements
  - If the condition is true then execute some instructions
  - else (the condition is false) execute another set of instructions
- Multi-way Decisions
  - Nested If-Then or Nested If-Then-Else Statements
  - Many options…

If Statements

- If statements take different forms
  - For now we will focus on the 2 basic forms
    - If-Then
    - If-Then-Else
  - Both of these statements can be nested
- A simple "if-then statement" is a one-way stmt
  - One-way decisions
    - If a condition is true → execute some special instructions

```
Syntax:
if (boolean expression)
{
   statements;
}
```


If-Then Statement




```
Syntax:
if (boolean expression)
{
   statements;
}
```

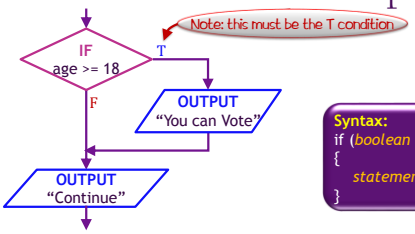
1. The Boolean expression is evaluated
2. If it evaluates to TRUE, then the True Instructions are executed
3. If it evaluates to FALSE the statement is ignored and the program continues with the next executable statement

If – Then Statement

- If some condition is true then we execute some set of instructions
- Otherwise we don't do anything special

IF Joe is 18 or older THEN tell him he can vote

Exercise: If-Then Example




```
Syntax:
if (boolean expression)
{
   statements;
}
```

What would the output be for…

age= 18?

age = 16?

Nesting If-Then Statements

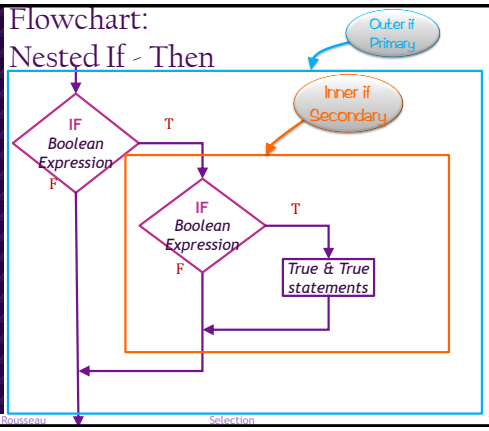
- If we need to check if more than one condition is true to execute some instructions
- Otherwise we don't do anything special

IF Joe is over 18 THEN
  IF Joe is registered THEN
    tell him he can vote.

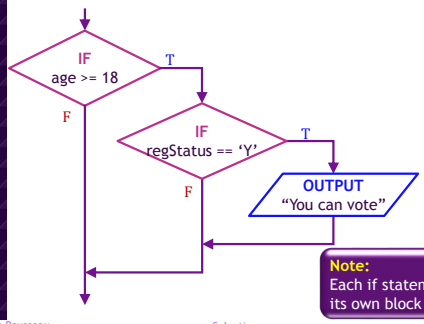
Nested If Statements

- Nested Selection Structure
  - Selection structure within another selection structure
  - Used when more than one decision must be made before an appropriate action can be carried out
- Primary Decision
  - always made by the outer selection structure
- Secondary Decision
  - Always made in the inner (or nested) selection structure
- Nested If-Then statement
  - an If-Then statement that contains another If-Then statement (within the statement section)

Flowchart: Nested If - Then



Example: Nested If-Then

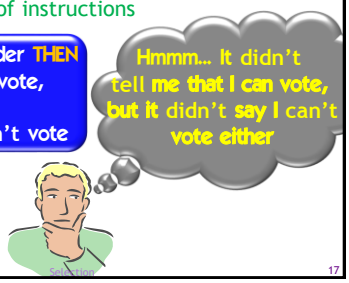


Note:
Each if statement has its own block {}

If-Then-Else Statements

- If some condition is true then we execute some set of instructions
- else (when the condition is false) we execute a different set of instructions

IF Joe is 18 or older THEN
  tell him he can vote,
ELSE
  tell him he can't vote



If-Then-Else Statements

Two-way Decisions

- Either execute one set of instructions or another
- Based on a Boolean expression

If the condition is true then

- Execute one set of instructions

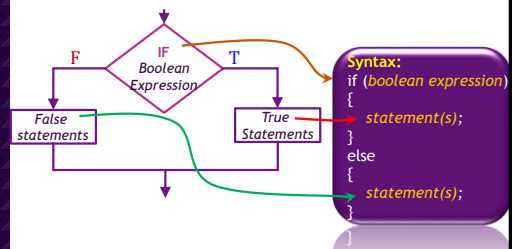
Else

- Execute another set of instructions

Syntax:

```
if (boolean expression)
{
   statement(s);
}
else
{
   statement(s);
}
```

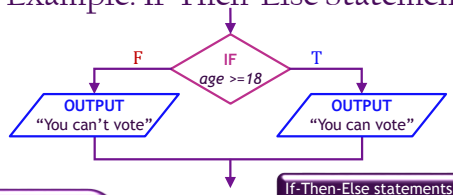
Flowchart for If-Then-Else Stmt




```
Syntax:
if (boolean expression)
{
   statement(s);
}
else
{
   statement(s);
}
```

1. The Boolean expression is evaluated
2. If it evaluates to TRUE, then statement1 is executed
3. If it evaluates to FALSE, then statement2 is executed

Example: If-Then-Else Statement



If-Then-Else statements can also be nested

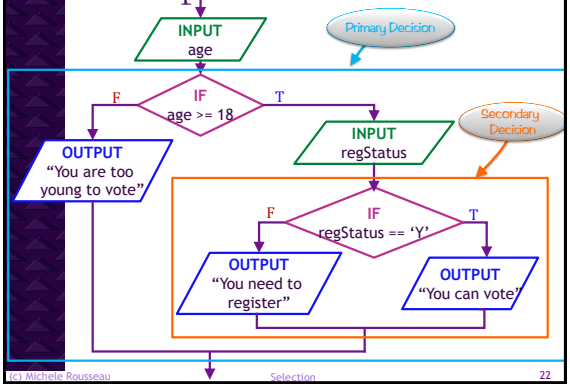
```
Syntax:
if (boolean expression)
{
   statement(s);
}
else
{
   statement(s);
}
```


Nesting If-Then-Else Statements

- If we need to check if more than one condition is true to execute some instructions
- Otherwise we don't do anything special

```
IF Joe is over 18 THEN
   IF Joe is registered THEN
      Tell him he can vote
   ELSE
      Tell him he needs to register
ELSE
   Tell him he is too young to vote
```

Example: Nested If-Then-Else



Example: Nested If-Then-Else

Nesting IF statements

- We could have nested on the false side of the if-then-else
- We could also nest more than one time
  - Given the syntax…
  - We can put any statement in between the brackets { }
  - An *if statement* is just another statement so it too can go between the brackets

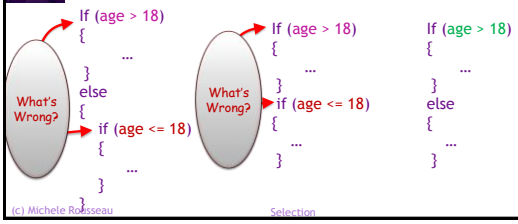
```
Syntax:
if (boolean expression)
{
   statement(s);
}
else
{
   statement(s);
}
```

Common Errors in Selection Structures

- Syntax errors:
  - Forgetting the parenthesis
    - Eg. if (age >= 18) → NOT if age >= 18
  - Putting a ";" at the end of the first line
    - Eg. if (age >= 18) → NOT if (age >= 18);
    - The statement is correct it just won't do anything

Common Logic Errors in If Structures

- Logic errors are commonly made as a result of the following mistakes:
  - Reversing the primary and secondary decisions
  - Redundancy
    - Using an unnecessary nested selection structure
  - Using if-then instead of if-then-else



Comparing c-strings

Cstrings are stored in an array

- Remember an array is a contiguous area of storage where each element has the same data type
- cstrings are an array of characters

- When you access an array you are working with the address of the array → Not the value in the address
- When you make the following comparison:

  if(stringOne == stringTwo)

you are comparing the addresses →not the values
→ the addresses will never be the same

strcmp function

strcmp(*c-string1*, *c-string2*);

Compares the contents of string1 and string 2
Returns:

Return value	if ASCII VALUES are such that
0	string1 == string2
Integer < 0	string1 < string2
Integer > 0	string1 > string2

Example

```
char stringOne[10];
char stringTwo[10];

cout << "Enter the first string: ";
cin   >> stringOne;

cout << "Enter the second string: ";
cin   >> stringTwo;

if(strcmp(stringOne, stringTwo) == 0)
{
   cout << "The strings are the same";
}
```


Assigning C-Strings

Similarly, we can't directly assign one c-string into another

For example:
char name1[30];
char name2[30];
We can't do this: name1 = "Joe";

Instead we can use:

strncpy(*toC-string*, *fromC-string2* , *sizeOfToC-string*);

```
strncpy(name1, "Joe", 30);
cout  << name1;
```

This will output: Joe

```
strncpy(name2, "Mo", 30);
strncpy(name1, name2, 30);
cout  << name1;
```

This will output: Mo

NOTE: You need to #include <cstring> to use strncpy

Use Constants for C-String Sizes

Similarly, we can't directly assign one c-string into another

For example:

```
const int NAME_SIZE = 30;
char name1[NAME_SIZE];
char name2[NAME_SIZE];

cout << "Enter your name: ";
cin.getline(name1, NAME_SIZE);

strncpy(name2, name1, NAME_SIZE);
cout << name1 << '\t'
     << name2;
```

NOTE: Using a constant for the size is a best practice. It is less error prone and increases modifiability