Programs with Branching Structure (wrapping up if-else, switch statement)

ESC101: Fundamentals of Computing

Nisheeth

Recap: Various ways of using if and else

```
if (condition) {
}
```

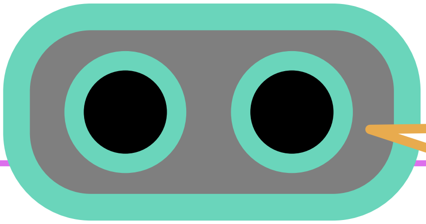
```
if (condition) {
}
else {
}
```

```
if (condition-1) {
}
else if (condition-2) {
}
else {
}
```

```
if (condition-1) {
}
else if (condition-2) {
}
else if (condition-3) {
}
        ⋮
else if (condition-N) {
}
else {
}
```

```
if (condition-1) {
    if (condition-2) {
    }
    else {
    }
}
else {
    if (condition-3) {
    }
    else {
    }
}
```

“nested” if



Note: Each else must have a matching if (also, number of if must be equal to or more than number of else)

Be Careful with Braces when using if-else

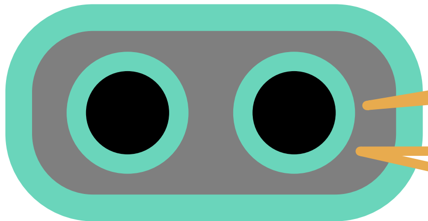
If you do not put curly braces, Mr. C will try to put them for you (and maybe in a way that you don't want him to)

If you write like this….

```
if((a != 0) && (b != 0))
    if(a * b >= 0)
        printf("Positive product");
else
    printf("One number is zero");
```

Mr. C will treat it like this internally

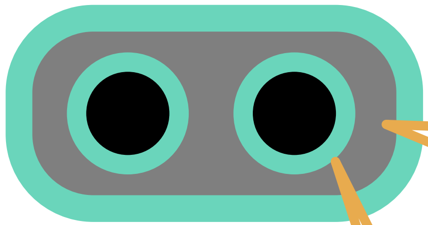
```
if((a != 0) && (b != 0)){
    if(a * b >= 0){
        printf("Positive product");
    }else{
        printf("One number is zero");
    }
}
```


If you do not put brackets, I will match else to closest if

I will not care how you did indentation

But that is not what I meant

One Last If-Else Example



```
5 6 7
```

```c
#include<stdio.h>
 int main() {
        int i = 5, j = 6, k = 7;
        if(i > j == k)
                printf("%d %d %d", i++, ++j, --k);
         else
               printf("%d %d %d", i, j, k);
         return 0;
 }
```

Reason:
Left-to-right associativity of relational operators

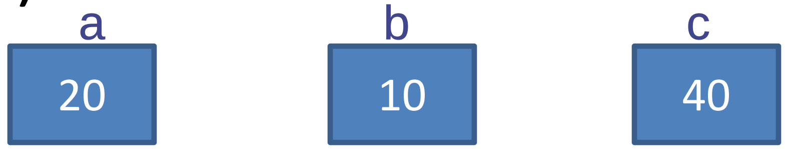
$(5 > 6) == 7$

$0 == 7$

0

Clarification: conditional operator associativity

- Associativity goes from right to left
- Applies only when there is more than one conditional operator to evaluate in an expression
  - – Does not affect the order of evaluation of expressions within the conditional operator (I think I said otherwise in the last class; that was not correct)



```
answer = a > b ?  a > c ? a: c  :  b > c ? b: c  ;
```

Reminder: Use Indentation..

- This is a main statement
    - This is a dependent statement

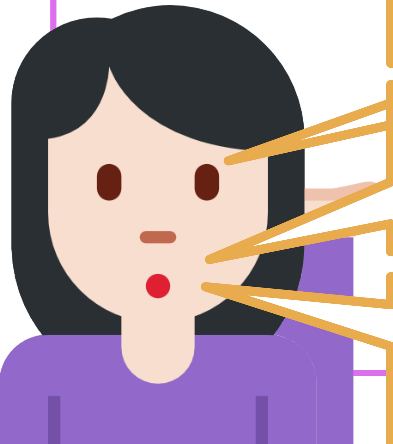
- Main statements are statements in the main control flow of your program
    - Dependent statements branch off from the main flow
    - Indent them, for easier understanding of code
    - Matters more in some languages, like Python

- Use 4 spaces instead of tab to indent

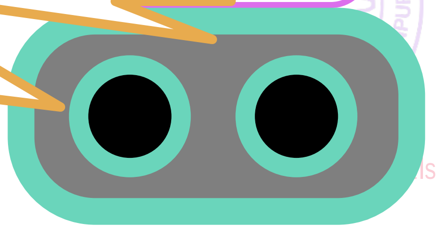
Print the name of the day of the week

```
if(n == 1) printf("Monday");
else if(n == 2) printf("Tuesday");
else if(n == 3) printf("Wednesday");
else if(n == 4) printf("Thursday");
else if(n == 5) printf("Friday");
else if(n == 6) printf("Saturday");
else if(n == 7) printf("Sunday");
```

```
switch(n){
    case 1: printf("Monday"); break;
    case 2: printf("Tuesday"); break;
    case 3: printf("Wednesday"); break;
    case 4: printf("Thursday"); break;
    case 5: printf("Friday"); break;
    case 6: printf("Saturday"); break;
    case 7: printf("Sunday"); break;
}
```



Just like if-else block is a single statement!
Still too much code – any shortcuts?
Sometimes not indenting looks neater



Yes, can use switch inside if,else
The switch statement
This whole *block* is one valid statement

Structure of Switch Statement

```
switch(integer expression){
    case label-1: ... break;
    case label-2: ... break;
    ...
    case label-n: ... break;
    default: ... break;
}
```

Must be an integer expression, e.g a, b+2, c*3 where a,b,c are int

Double, float expressions banned

Unless typecast to int

For relational expressions, Mr C will warn but work

case a+2: wrong label

Labels must be integer or character **constants**

Labels must not repeat

Labels can be in any order (not necessarily increasing/decreasing)

I'll give a warning but interpret 0, 1 as int

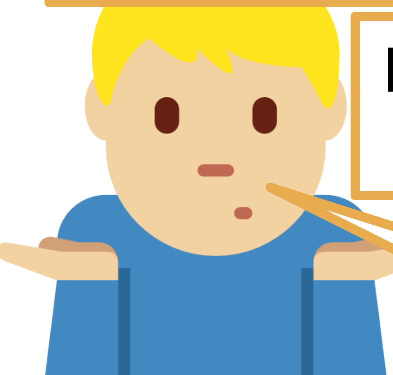
Can put any number of statements here, math formulae, printf, if-else, another switch (nested)

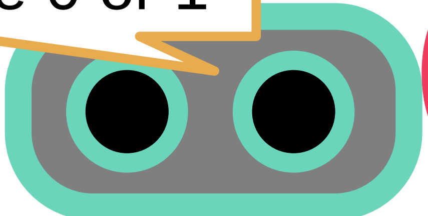
??

Relational expressions generate value 0 or 1

??

Careful about brackets





The Working of Switch Statement

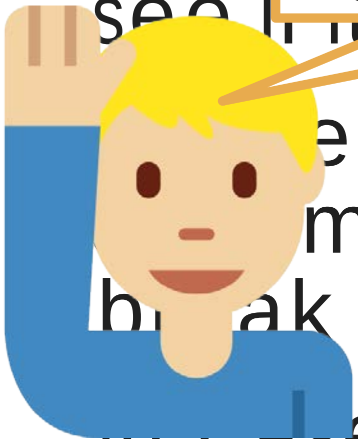
First, ... expr

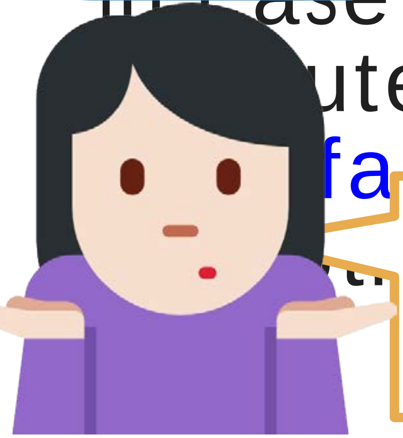
v is c ... s see i... qual to any one

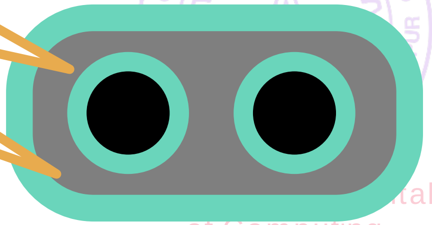
...el matches, execute ...ments next to it till b...ak is encountered

...se no label matches ...ute statements next ...fault (if no default, ...t

```
switch(integer expression){
    case label-1: ... break;
    case label-2: ... break;
    ...
    case label-N: ... break;
    default: ... break;
}
```


If we want to check for inequality or work with float etc, we can always write if-else statements ourselves




Switch-case is a shortcut that only checks for equality and that too only with integers
Exactly

The Default Case

The English word default can mean failure to fulfil a promise (*bank loan default*)

… or it can mean a rule that applies when no other rule applies

In switch case, whatever we write in default is executed if none of the labels match – used to handle incorrect input

Can put the default anywhere, not necessarily at end

Need not put default case at all. If we don’t put a default case, Mr C will do nothing if no labels match

The Break Statement

The switch case statement behaves in a funny manner

Mr C finds the label that matches (else default if none match) but keeps executing all statements (**even those of other labels and default**) till encounters a *break;*

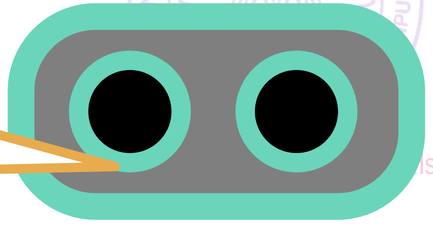
This behaviour is called *fall-through*

Once *break;* is encountered, Mr C claims he is done with the switch statement – *break;* stops Mr C's fall ☺

That is why no brackets needed

case 2: { ... } break;

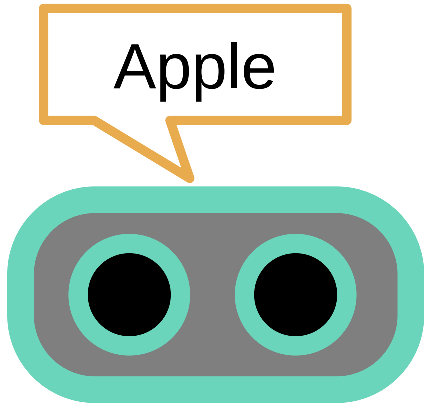
Not needed



Yes, the break; statement tells me when to stop

switch: Some More Examples

```
#include<stdio.h>
 int main() {
        char ch = 65;
        switch(ch) {
                case 'A': printf("Apple");
                break;
                case 'B': printf("Bing");
                break;
               default: printf("Bye");
                break;
         }
        return 0;
 }
```

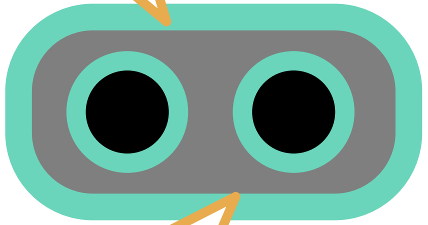


switch: Some More Examples

```
#include<stdio.h>
 int main() {
        char ch;

        scanf(“%c”,&ch);
        switch(ch) {
                case ‘a’:
                case 'A': printf("Apple");
                break;
                case ‘b’:
                case 'B': printf(“Banana");
                break;
                case ‘c’:
                case ‘C’: printf(“Cherry”);
                break;
                default: printf("Bye");
                break;
         }
        return 0;
 }
```

a or A both will print Apple
b or B both will print Banana
c or C both will print Cherry



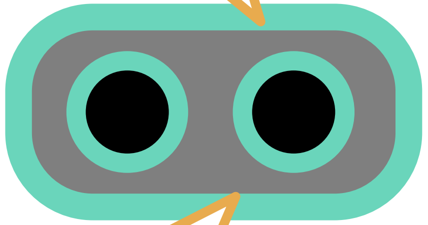
Without break; I will “fall through” all cases until I see break;

switch: Some More Examples

```
#include <stdio.h>
int main(){
   int n;
   scanf(“%d”,&n); // read the day number
   switch(n){
      case 2:
      case 3:
      case 4:
      case 5:
      case 6: printf("Weekday"); break;
      case 1:
      case 7: printf("Weekend"); break;
      default: printf("Illegal day"); break;
   }
```

If n is 2/3/4/5/6, will print Weekday

If n is 1 or 7, will print Weekend



Without break; I will “fall through” all cases until I see break;

switch vs if-else

- Some limitations of switch as compared to if-else
  - float expressions can’t be tested in switch
  - Can’t use variables for case labels
- Advantages of switch over if-else
  - switch is much faster than if-else
  - Reason: Compiler creates a “jump table” for switch internally. In contrast, if-else conditions are evaluated at run-time (thus slower especially if the conditions are very complex)
- But we now know both. ☺ Can even mix-and-match if-else and switch

A Small Quiz

- What will the following piece of code do?

```
(5<2) && (3/0)
```

- Compile error ?
- Run-time error ?
- Output 1 ?
- Output 0 ?

Short-circuit evaluation of Logical Operators

- Mr. C does not evaluate the second operand of **binary logical operator** if the final result can be deduced from **first operand**

$$\underbrace{(5<2)}_{0} \;\&\&\; (3/0) \qquad \text{Result} = 0$$

- Now answer what will the output of the following?

$$\overset{1}{!}(\underbrace{(2>5)}_{0} \;\overset{0}{\&\&}\; (3/0)) \;||\; (4/0) \qquad \text{Result} = 1$$

A Large Quiz

- Coming up next Wednesday
- Syllabus
  - everything covered up to today
- Logistics
  - In class, during class hours on Wednesday, 29th January
  - Please be in your seat at noon
  - Ok to bring one sheet of paper with notes on it
  - Please don't bring cell phones to the class that day