CS50 - SECTION 1

9/18/18

About Me

My name is Wesley De Silvestro. Call me **Wes**.

- I’m a sophomore in Winthrop House.
- I’m from Lakeland, Florida.
- I took CS50 last fall.
- I study government and enjoy integrating approaches from CS and statistics.
- I can’t wait to experience CS50 with you all this semester!

ROUNDTABLE:

INTRODUCTIONS

(name, year, what your Scratch project was, one thing you want to get out of CS50)

HOW TO CS50

GOALS OF THE COURSE

1. Learn to think computationally and algorithmically.
2. Improve your problem-solving skills—this will help you in every other course and in real-life too!
3. Gain practical programming skills in a plethora of languages (C, Python, JavaScript, etc.)
4. Familiarize yourself with best practices for software design.
5. Build a community with your peers that will stay with you after this course.

COMPONENTS TO CS50

1. Lectures/Section
2. Problem Sets
3. Quiz
4. Final Project

You'll spend ~75% of your time in the class here.

HOW TO SUCCEED? - A WEEK IN THE LIFE

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
	Section		Problem Set Due	Lecture		
- Continue work and visit OH as needed	- Attend section with questions ready	- Final touches to problem set and try to submit today!		- Attend lecture - Look over the problem set	- Work on the problem set	- Continue work and visit OH as needed

WHAT RESOURCES ARE THERE?

1. Notes from lecture
2. Office hours - https://cs50.harvard.edu/2018/fall/hours/
3. Walkthrough videos on problem sets
4. “Shorts” and practice problems provided in Section
5. CS50 Discourse - https://discourse.cs50.net
6. Your peers (with respect to the collab. policy)
7. Section

WHAT TO EXPECT FROM SECTION

- Post-mortem of previous problem set
- Review of concepts from lecture
  - **Each week, I expect you've watched the previous week's lecture, read the problem set specification, and are ready with questions**
- Practice problems for hands-on experience
- Workshops (e.g. debugging, improving program design, etc.)
- *What you all request*
  - Send me an email ~24 hours before section if there is something you want to cover

A QUICK NOTE

- You <u>have</u> to attend section, but it doesn’t have to be this one
  - Just make sure you get marked off for attendance each week
- Tip: Shop different sections and find the one that best fits your learning style

GETTING IN CONTACT WITH ME

wadesilvestro@college.harvard.edu

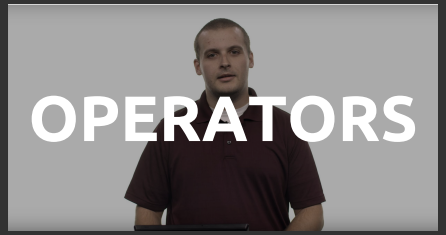
(863) 666-4003

CONCEPTS DEEP-DIVE

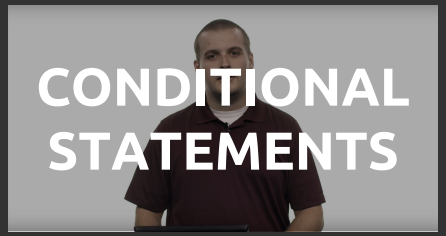
CHECK OUT THE “SHORTS”

https://www.youtube.com/watch?v=lnYKOnz9ln8

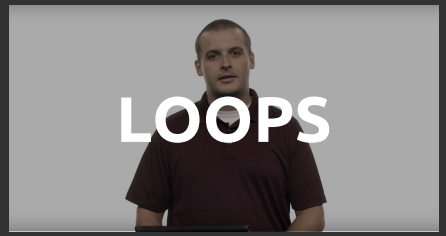
https://www.youtube.com/watch?v=q6K8KMqt8wQ



https://www.youtube.com/watch?v=7apBtlEkJzk



https://www.youtube.com/watch?v=FqUeHzvci10



https://www.youtube.com/watch?v=QOvo-xFL9II

VARIABLES - THE BASICS

```
int main(void)
{
    int x = 14;

    int y;
    y = 14;
}
```

What is the difference between these two approaches?

VARIABLES - THE BASICS

```
int main(void)
{
    int x = 14;

    int y;
    y = 14;
}
```

What is the difference between these two approaches?

In the first one, we declare and initialize a variable at the same time (*instantiation*). In the second one, we separate these operations.

VARIABLES - TYPING

C is a **strongly typed** language - Every variable you declare must include a type associated with it.

What is the purpose for this? What advantages and disadvantages does this confer?

VARIABLES - TYPING

C is a **strongly typed** language - Every variable you declare must include a type associated with it.

What is the purpose for this? What advantages and disadvantages does this confer?

VARIABLES - PRIMITIVES

DATA TYPE	RANGE	MEMORY
`int`	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647	2 or 4 bytes
`char`	-128 to 127 or 0 to 255	1 byte
`long`	-2,147,483,648 to 2,147,483,647	4 bytes
`float`	1.2E-38 to 3.4E+38	4 bytes
`double`	2.3E-308 to 1.7E+308	8 bytes

VARIABLES - PRIMITIVES

DATA TYPE	RANGE	MEMORY
`int`	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647	2 or 4 bytes
`char`	-128 to 127 or 0 to 255	1 byte
`long`	-2,147,483,648 to 2,147,483,647	4 bytes
`float`	1.2E-38 to 3.4E+38	4 bytes
`double`	2.3E-308 to 1.7E+308	8 bytes

?

VARIABLES - PRIMITIVES

DATA TYPE	RANGE	MEMORY
`int`	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647	2 or 4 bytes
`char`	-128 to 127 or 0 to 255	1 byte
`long`	-2,147,483,648 to 2,147,483,647	4 bytes
`float`	1.2E-38 to 3.4E+38	4 bytes
`double`	2.3E-308 to 1.7E+308	8 bytes

?

?

VARIABLES - QUALIFIERS

- The `unsigned` qualifier means you have NO signs on your data type (i.e. you can't have negatives)
- The `signed` qualifier allows you to have signs

VARIABLES - QUALIFIERS

- The `unsigned` qualifier means you have NO signs on your data type (i.e. you can't have negatives)
- The `signed` qualifier allows you to have signs

How might we find the size of a data type in C?

Arithmetic Operators

Let’s brainstorm together: What arithmetic operators do we have in C?

Arithmetic Operators

Let’s brainstorm together: What arithmetic operators do we have in C?

+ - * / %

Arithmetic Operators - Shortcuts

Take advantage of shortcuts:

```
int main(void)
{
    int x;

    x = x + 1;
    x += 1;
    x++;
}
```

Other Operators

TYPE	OPERATOR	PURPOSE
Logical	&& || !	AND OR NOT
Relational	< <= >= > == !=	LESS THAN LESS THAN OR EQUAL TO GREATER THAN OR EQUAL TO GREATER THAN EQUALS NOT EQUAL TO

Other Operators

TYPE	OPERATOR	PURPOSE
Logical	&& || !	AND OR NOT
Relational	< <= >= > == !=	LESS THAN LESS THAN OR EQUAL TO GREATER THAN OR EQUAL TO GREATER THAN EQUALS NOT EQUAL TO

What's the difference between = and == in C?

MORE ON OPERATORS

What is `!(P && Q)` equivalent to?

MORE ON OPERATORS

What is `!(P && Q)` equivalent to?

`!P || !Q`

How about `!(P || Q)`?

MORE ON OPERATORS

What is `!(P && Q)` equivalent to?

`!P || !Q`

These are called De Morgan's Laws.

How about `!(P || Q)`?

`!P && !Q`

CONDITIONAL STATEMENTS

How do we actually use those logical operators we just learned?

CONDITIONAL STATEMENTS

How do we actually use those logical operators we just learned?

By writing boolean expressions and utilizing conditional statements!

```
if (condition)
   {
       // code goes here if the
condition is met
   }
```

```
if (condition)
   {
       // code goes here
   }
else
   {
       // if condition fails
   }
```

```
if (condition)
   {
       // code goes here
   }
else if (another condition)
   {
       // if condition fails
   }
else
   {
```

CONDITIONAL STATEMENTS - PRACTICE

How might I write a statement that prints “hello, world” to the screen only if the user enters an integer greater than 10?

CONDITIONAL STATEMENTS - PRACTICE

How might I write a statement that prints “hello, world” to the screen only if the user enters an integer greater than 10?

```
#include <cs50.h>
#include <stdio.h>

int main(void)
{
   int x = get_int("Please enter an integer: ");

   if(x > 10) {
       printf("hello, world\n");
   }
}
```

THE SWITCH STATEMENT

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

  /* local variable definition */
  char grade = 'B';

  switch(grade) {
     case 'A' :
        printf("Excellent!\n" );
        break;
     case 'B' :
        printf("Well done\n" );
        break;
     case 'C' :
        printf("You passed\n" );
        break;
     case 'D' :
        printf("Better try again\n" );
        break;
     default :
        printf("Invalid grade\n" );
  }

  printf("Your grade is  %c\n", grade );
  return 0;
}
```

THE SWITCH STATEMENT

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

  /* local variable definition */
  char grade = 'B';

  switch(grade) {
     case 'A' :
        printf("Excellent!\n" );
        break;
     case 'B' :
        printf("Well done\n" );
        break;
     case 'C' :
        printf("You passed\n" );
        break;
     case 'D' :
        printf("Better try again\n" );
        break;
     default :
        printf("Invalid grade\n" );
  }

  printf("Your grade is  %c\n", grade );
  return 0;
}
```

Must be constant values like a character or number

THE SWITCH STATEMENT

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

  /* local variable definition */
  char grade = 'B';

  switch(grade) {
     case 'A' :
        printf("Excellent!\n" );
        break;
     case 'B' :
        printf("Well done\n" );
        break;
     case 'C' :
        printf("You passed\n" );
        break;
     case 'D' :
        printf("Better try again\n" );
        break;
     default :
        printf("Invalid grade\n" );
  }

  printf("Your grade is  %c\n", grade );
  return 0;
}
```

Must be constant values like a character or number

What does the break statement do?

THE SWITCH STATEMENT

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

  /* local variable definition */
  char grade = 'B';

  switch(grade) {
     case 'A' :
        printf("Excellent!\n" );
        break;
     case 'B' :
        printf("Well done\n" );
        break;
     case 'C' :
        printf("You passed\n" );
        break;
     case 'D' :
        printf("Better try again\n" );
        break;
     default :
        printf("Invalid grade\n" );
  }

  printf("Your grade is  %c\n", grade );
  return 0;
}
```

The break statement tells C to exit the switch statement. Without it, the program would flow into the other cases and execute their code as well.

THE SWITCH STATEMENT

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

  /* local variable definition */
  char grade = 'B';

  switch(grade) {
     case 'A' :
        printf("Excellent!\n" );
        break;
     case 'B' :
        printf("Well done\n" );
        break;
     case 'C' :
        printf("You passed\n" );
        break;
     case 'D' :
        printf("Better try again\n" );
        break;
     default :
        printf("Invalid grade\n" );
  }

  printf("Your grade is  %c\n", grade );
  return 0;
}
```

When would we want to use a switch statement vs. just if-else statements?

CONDITIONAL STATEMENTS - A SHORTCUT

We have the **ternary operator** in C to allow us to shorten our if-else statements:

```
if (a > b) {
   result = x;
}
else {
   result = y;
}
```

```
result = a > b ? x : y;
```

COMPOUND BOOLEAN EXPRESSIONS

How do we express two different boolean conditions?

COMPOUND BOOLEAN EXPRESSIONS

How do we express two different boolean conditions?

Using our logical operators! For example:

```
((x > 15) && (y == 7))
```

COMPOUND BOOLEAN EXPRESSIONS

Note that C reads left-to-right like in English:

```
((x > 15) && (y == 7) || (z > 12))
```

This will calculate the boolean expression off the first two parts and then compare that result with the `||` to the last part.

LOOPS

We have three types of loops in C:

- `while` loops
- `do-while` loops
- `for` loops

What is the difference between each of these?

LOOPS

while	do-while	for
Checks for a condition at the start of each time the loop runs	Runs the code in the loop body and then checks the condition	Checks a condition prior to running the loop, runs the code in the loop body, increments the counter, and then repeats

`int i = 0;`
`while (i < 10)`
`{`
`  printf("%i\n", i);`
`  i++;`
`}`
	`int j = 0;`
	`do`
	`{`
	`  printf("%i\n", j);`
	`  j++;`
	`}`
	`while (j < 10);`
		`for (int k = 0; k < 10; k++)`
		`{`
		`  printf("%i\n", k);`
		`}`

LOOPS

When do we use each different type of loop?

LOOPS

When do we use each different type of loop?

`while`	`do-while`	`for`
You want to run the code in the loop body until there is a state change (*nondeterministic*)	You want to run the code in the loop body until there is a state change, but guarantee the code runs <u>at least once</u>	You want to run the code in the loop body for a predetermined number of times (*deterministic*)

LOOPS

When do we use each different type of loop?

`while`	`do-while`	`for`
You want to run the code in the loop body until there is a state change (*nondeterministic*)	You want to run the code in the loop body until there is a state change, but guarantee the code runs <u>at least once</u>	You want to run the code in the loop body for a predetermined number of times (*deterministic*)

You can see the for loop as a subset of the while loop

HANDS-ON PRACTICE

PAIR PROBLEM #1: GRADES

http://bit.ly/2D4qfCj

PAIR PROBLEM #1: GRADES [TASK 1]

```
#include <cs50.h>
#include <stdio.h>

int main(void)
{
   int n = get_int("Enter a valid grade (between 60 and 100): ");
   if (n >= 90)
       printf("You got an A!\n");
   else if (n >= 80)
       printf("You got a B!\n");
   else if (n >= 70)
       printf("You got a C!\n");
   else
       printf("You got a D!\n");
}
```

PAIR PROBLEM #1: GRADES [TASK 2]

```
#include <cs50.h>
#include <stdio.h>

int main(void)
{
   char n = get_char("Enter a valid grade letter: ");
   switch(n)
   {
       case 'A':
           printf("You got between 90-100.\n");
           break;
       case 'B':
           printf("You got between 80-89.\n");
           break;
       case 'C':
           printf("You got between 70-79.\n");
           break;
       case 'D':
           printf("You got between 60-69.\n");
           break;
       case 'F':
           printf("You got between 0-59.\n");
           break;
       default:
           printf("Please enter a valid grade (A, B, C, D, or F).\n");
   }
}
```

PAIR PROBLEM #2: MULTIPLES

http://bit.ly/2Nm60EE

PAIR PROBLEM #2: MULTIPLES

```
#include <cs50.h>
#include <stdio.h>

int main (void)
{
   int input = get_int("Please enter a number between 1 and 100: ")

   for(int i = 1; i <= 100; i++)
   {
       if (i % input == 0)
           printf("%d  ", i);
   }
   printf("\n");
}
```

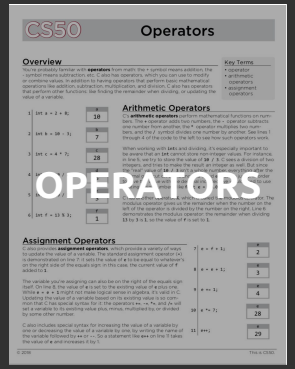
PROBLEM SET 1 PREVIEW

PROBLEM SET 1 PREVIEW

You will need to complete:

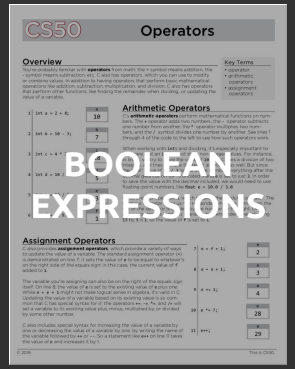
- **`hello.c`**
- One of the two **`mario.c`** options (less/more)
- Either **`cash.c`** or **`credit.c`**

REFERENCE SHEETS



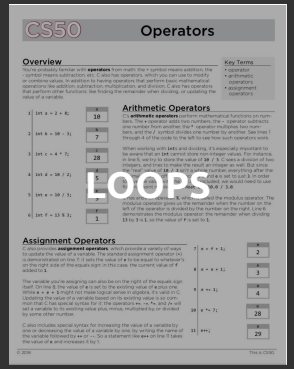
OPERATORS

https://www.dropbox.com/sh/5y662ey1hc4sde4/AABpC6MbC5rzo81wNK9CP_NZa/Operators.pdf?dl=0



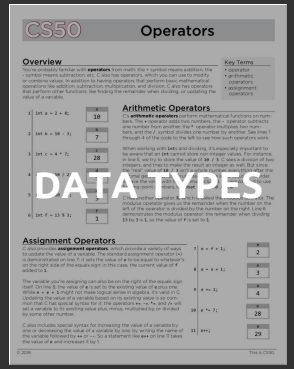
BOOLEAN EXPRESSIONS

https://www.dropbox.com/sh/5y662ey1hc4sde4/AAD30Rk8Fcmz_HG2hE-_bXoRa/Boolean%20Expressions.pdf?dl=0



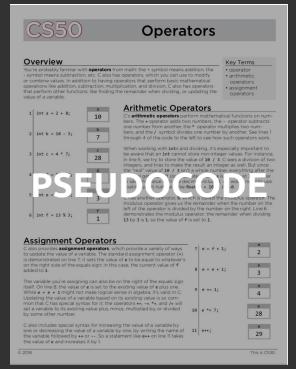
LOOPS

https://www.dropbox.com/sh/5y662ey1hc4sde4/AAA3J_QHkJ5GFeTi2YuEJplYa/Loops.pdf?dl=0



DATA TYPES

https://www.dropbox.com/sh/5y662ey1hc4sde4/AAC10N2PXZrLldLKZz21hCp2a/Data%20Types.pdf?dl=0



PSEUDOCODE

https://www.dropbox.com/sh/5y662ey1hc4sde4/AAAac4DxJ3fRQiaohQ3dts65a/Pseudocode.pdf?dl=0

QUICK SURVEY

Survey Link

You're free to go once you complete this:

https://bit.ly/2xkI2zX