This is CS 50.

Harvard College's Introduction to Computer Science I

COMPUTER SCIENCE 50

WEEK 1

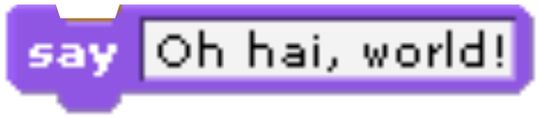
DAVID J. MALAN '99
malan@post.harvard.edu

Scratch Meets C


```
int
main(int argc, char *argv[])
{
    printf("O hai, world!\n");
}
```

Statements

Scratch v. C




```
printf("O hai, world!\n");
```

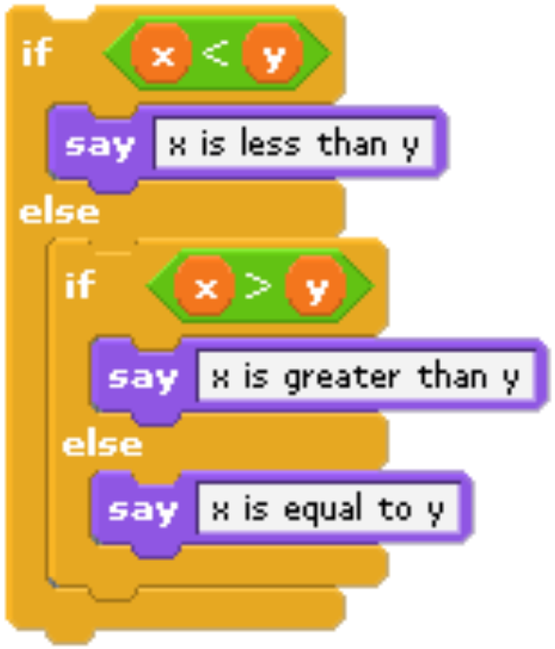
Boolean Expressions

Scratch v. C


```
(x < y)
((x < y) && (y < z))
```

Conditions

Scratch v. C




```
if (x < y)
{
    printf("x is less than y\n");
}
else if (x > y)
{
    printf("x is greater than y\n");
}
else
{
    printf("x is equal to y\n");
}
```

Loops

Scratch v. C


```
while (1)
{
    printf("O hai!\n");
}
```


```
for (int i = 0; i < 10; i++)
{
    printf("O hai!\n");
}
```

Variables

Scratch v. C

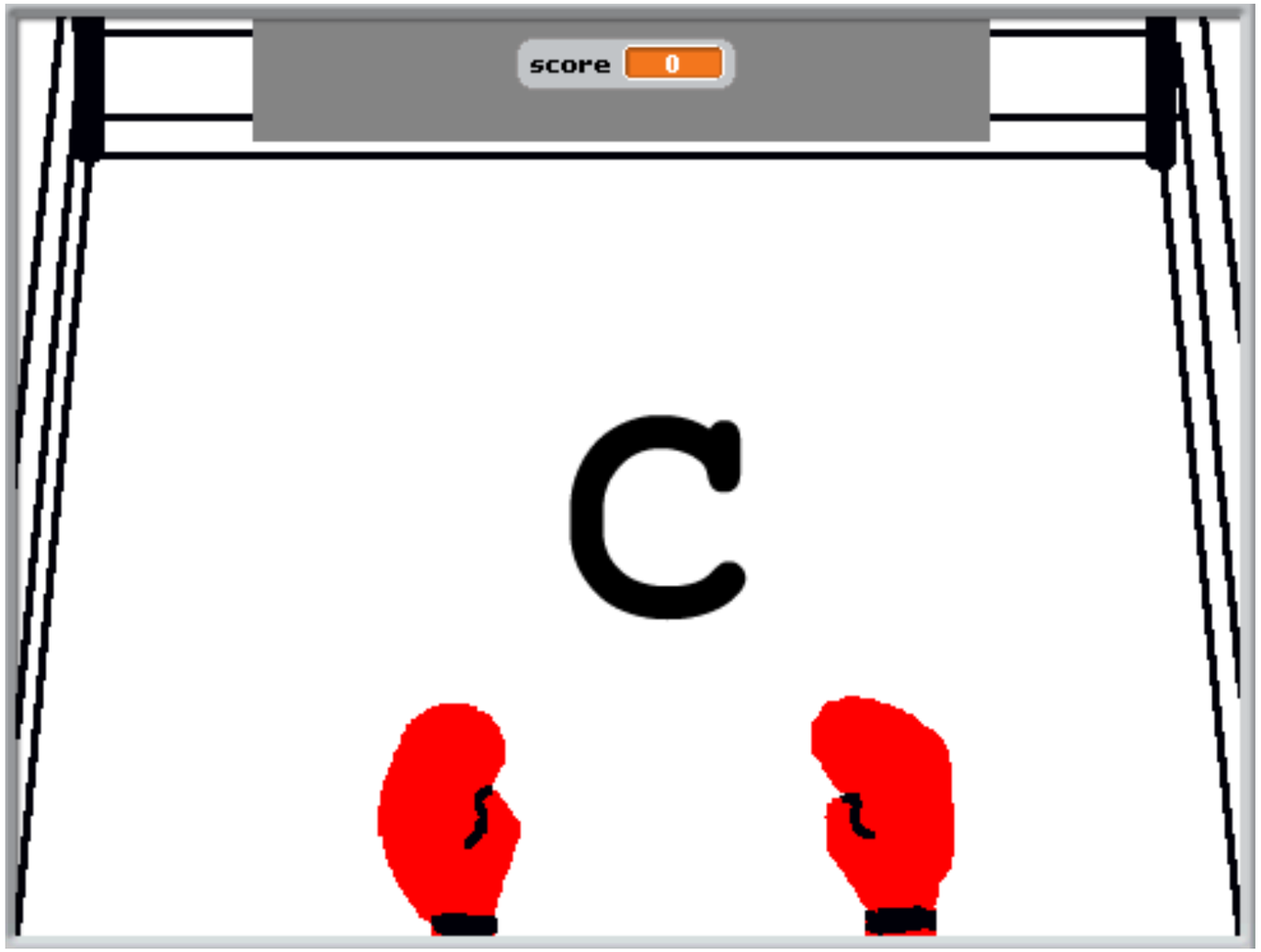

```
int counter = 0;
while (1)
 {
     printf("%d\n", counter);
     counter++;
}
```

Arrays

Scratch v. C


```
char *inventory[SIZE];
inventory[i] = "Orange";
```

Scratch versus C



O hai, C!

```c
#include <stdio.h>

int
main(int argc, char *argv[])
{
    printf("o hai, world!\n");
}
```

```
10000011 00000001 00010001 00000000 00111101 11111100 01110100 00111101
00000000 01000000 00000000 00000000 00000000 00000000 00000000 00000000
10010000 00000000 00000000 00000000 01010000 00000000 00000111 00110000
00001011 00000001 00001011 00000011 00001010 00000000 00000000 00000000
00000000 00100000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00100000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01110000 00010000 00000000 00100000 00000001 00000000 00000000 00000000
00000000 00000000 00000000 00100000 00000001 00000000 00000000 00000000
00000000 00000000 00000000 01000000 00000001 00000000 00000000 00000000
00000000 00100000 00000000 01000000 00000001 00000000 00000000 00000000
11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111
10010000 10000000 00000000 01000000 00000001 00000000 00000000 00000000
00101110 01100100 01111001 01101110 01100001 01101101 01101001 01100011
10110000 00000100 00000000 00100000 00000001 00000000 00000000 00000000
10110000 00000100 00000000 00100000 00000001 00000000 00000000 00000000
10100000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
10110000 00000100 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00100000 00000000 00000000
[...]
```

nice.fas.harvard.edu

cloud.cs50.net

Some Commands

- `cd`
- `cp`
- `ls`
- `mkdir`
- `mv`
- `pwd`
- `rm`

Some More Commands

- `cat`
- `ci`
- `co`
- `echo`
- `gcc`
- `gdb`
- `less`
- `make`
- `man`
- `more`
- `...`

How to Write a Program in C

```
1.  nano hai1.c
2.  gcc hai1.c
3.  a.out
```

see
hai1.c

How to Write a Program in C

(with a better name)

```
1.  nano hai1.c
2.  gcc -o hai1 hai1.c
3.  hai1
```

see
hai1.c

How to Write a Program in C

(with fewer keystrokes)

```
1.  nano hai1.c
2.  make hai1
3.  hai1
```

see
hai1.c

Editors

- Emacs
- Nano
- Vim
- ...

main

```
int main(int argc, char *argv[]);
```

Standard Output

printf

```
int printf(const char *format, ...);
```

see
http://www.cppreference.com/stdio/printf.html

Escape Sequences

- `\n`
- `\r`
- `\t`
- `\"`
- `\\`

see
http://www.cppreference.com/escape_sequences.html

Variables

Types†

- `char`
- `double`
- `float`
- `int`

† long, short, signed, unsigned

see
math1.c

Format Strings

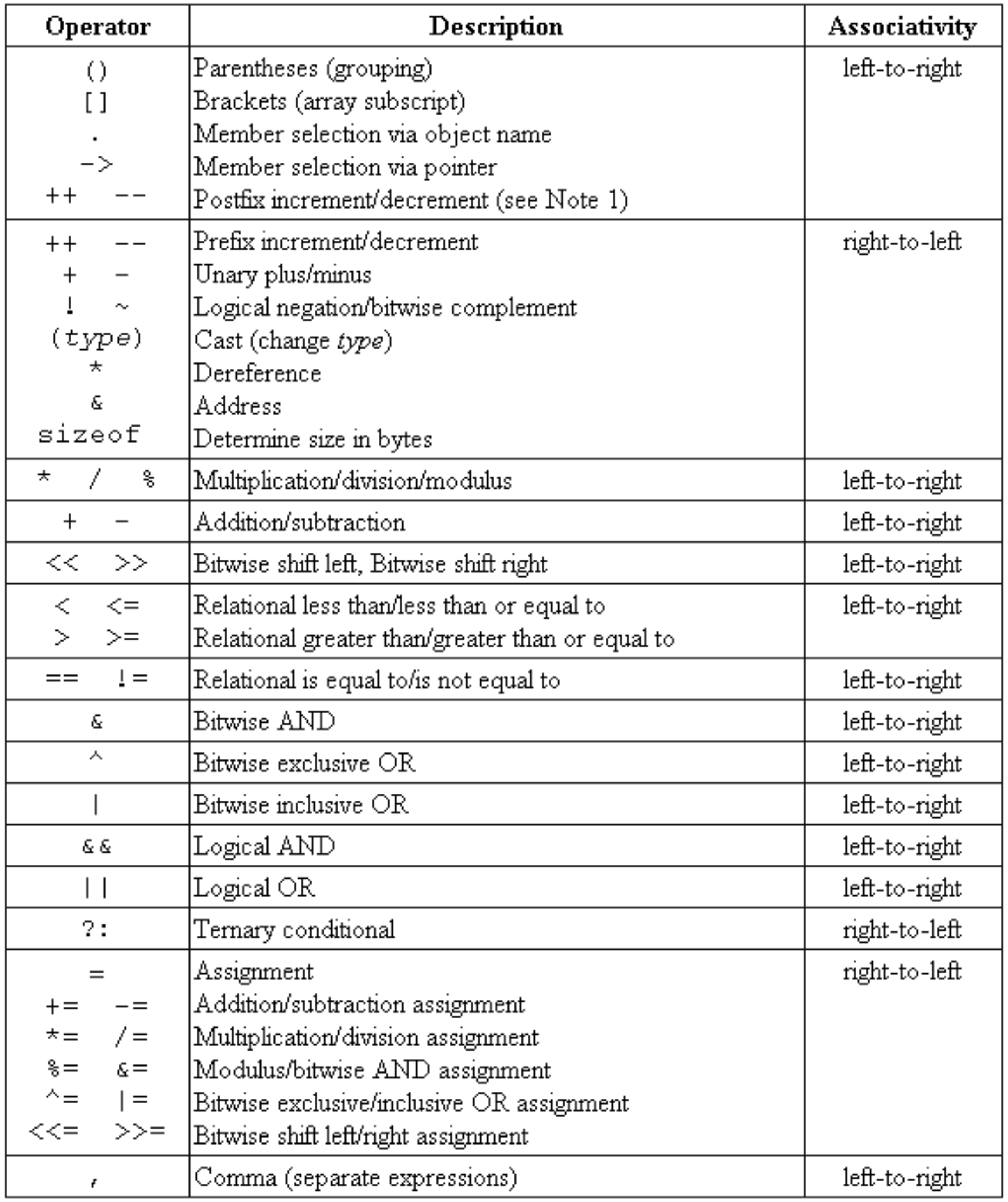
- %c
- %d
- %e
- %E
- %f
- %s
- %u
- %x

see
math2.c, sizeof.c
http://www.cppreference.com/stdio/printf.html

Arithmetic Operators

- +
- -
- *
- /
- %

Precedence



Operator	Description	Associativity

`()`
`[]`
`.`
`->`
`++ --`
	Parentheses (grouping)
	Brackets (array subscript)
	Member selection via object name
	Member selection via pointer
	Postfix increment/decrement (see Note 1)
		left-to-right

`++ --`
`+ -`
`! ~`
(*type*)
`*`
`&`
`sizeof`
	Prefix increment/decrement
	Unary plus/minus
	Logical negation/bitwise complement
	Cast (change *type*)
	Dereference
	Address
	Determine size in bytes
		right-to-left

`* / %`	Multiplication/division/modulus	left-to-right
`+ -`	Addition/subtraction	left-to-right
`<< >>`	Bitwise shift left, Bitwise shift right	left-to-right
`< <=` `> >=`	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
`== !=`	Relational is equal to/is not equal to	left-to-right
`&`	Bitwise AND	left-to-right
`^`	Bitwise exclusive OR	left-to-right
`|`	Bitwise inclusive OR	left-to-right
`&&`	Logical AND	left-to-right
`||`	Logical OR	left-to-right
`?:`	Ternary conditional	right-to-left

`=`
`+= -=`
`*= /=`
`%= &=`
`^= |=`
`<<= >>=`
	Assignment
	Addition/subtraction assignment
	Multiplication/division assignment
	Modulus/bitwise AND assignment
	Bitwise exclusive/inclusive OR assignment
	Bitwise shift left/right assignment
		right-to-left

`,`	Comma (separate expressions)	left-to-right

Chart a http://www.difranco.net/cop2220/op-prec.htm.

Width and Precision †

- %<width>.<precision>e
- %<width>.<precision>E
- %<width>.<precision>f
- %<width>.<precision>s

† -, +

see
math{3,4,5}.c
http://www.cppreference.com/stdio/printf.html

Variables

Types†

- `bool`
- `string`

† These are CS 50-specific.

see
`hai2.c`

Standard Input †

- `char GetChar();`
- `double GetDouble();`
- `float GetFloat();`
- `int GetInt();`
- `long long GetLongLong();`
- `string GetString();`

† These are CS 50-specific.

see
`hai3.c`, `adder.c`

How to Write a Program

(using CS 50's library)

```
1.  nano hai3.c
2.  gcc -o hai3 hai3.c -lcs50
3.  hai3
```

see
hai3.c

Fahrenheit to Celsius

$C = (5/9) \times (F - 32)$

```
#include <cs50.h>

#include <stdio.h>



int

main(int argc, char *argv[])

{





}
```

Conditions

if

```
if (condition)
{
    // do this
}
```

Conditions

if-else

```
if (condition)
{
    // do this
}
else
{
    // do that
}
```

see
conditions1.c

Conditions

if-else else-if

```
if (condition)
{
    // do this
}
else if (condition)
{
    // do that
}
else
{
    // do this other thing
}
```

see
conditions2.c

Boolean Expressions

```
if (condition || condition)
{
    // do this
}
else
{
    // do that
}
```

Boolean Expressions

```
if (condition && condition)
{
    // do this
}
else
{
    // do that
}
```

see
nonswitch.c

Conditions

switch

```
switch (expression)
{
    case i:
        // do this
        break;
    case j:
        // do that
        break;
    default:
        // do this other thing
}
```

see
switch{1,2}.c

Loops

for

```
for (initializations; condition; updates)
{
    // do this again and again
}
```

see
progress{1,2}.c

Loops

while

```
while (condition)

{

    // do this again and again

}
```

see
progress3.c

Loops

do-while

```
do
{
    // do this again and again
}
while (condition);
```

see
positive{1,2,3}.c