LEC 03

CSE 121

Strings, char, and Variables

BEFORE WE START

Talk to your neighbors:

What is your favorite emoji? 😀

Respond on sli.do!

Music: 🌻 CSE 121 26sp Lecture Tunes 🌻

Instructor: Matt Wang

TAs:

Abdul	Amogh	Anant	Anum	Cayden
Dalton	Ethan	Hayden	Jesse	Jessica
Johnathan	Minh	Navya	Paul	Reese
Ruslana	Sam	Savannah	Spencer	Shayna
Tamsyn	TJ	Trey		

Questions during Class?

Raise hand or send here

sli.do #cse121

Agenda (1/4)

- **Announcements, Reminders (now)**
- C0 Reflection Recap
- Casting, More Variables and Operators!
- Strings and Characters Review
  - code example!

Announcements, Reminders

- **P0: Cornbear's Café released!**
  - due Tuesday, April 14th
- Expect C0 grades ~ 1 week from submission (we'll announce on Ed)
  - shortly after, your first ***resubmission*** cycle will open!
- Ed Shortcuts page! (and search bar)
- next Wednesday: guest lecture from Hayden!

Agenda (2/4)

- Announcements, Reminders
- **C0 Reflection Recap (now)**
- Casting, More Variables and Operators!
- Strings and Characters Review
  - code example!

Bountiful Butterflies

```
 ------------
|  Butterfly  |
 ------------


   /\      |\
  /  \     | \
 /    \    |  \
|      \\/|   \
 \     /00|    /
  /--|[--]\--\
 /   / [--] \   \
 \   / [-]  \  /
  //         \\
  0           0
```

```
A Sad Butterfly

       \     /
        \___/
_______|00 |_______
\      | ^ |      /
 \     |   |     /
  \    |   |    /
   \   |   |   /
    \  |   |  /
     \ |   | /
      \|___|/
```

```
~~~~~~ Butterfly ~~~~~~

/"""""""\   \   /   /"""""""\
\        \ (   ) /        /
  >         ||          <
 /          ||           \
 \,,,,,,,,,/V\,,,,,,,,,/
```

```
  -Butterfly-

 _   __
|  |   \    _*
|  |    \  / _*
|  |     \|/
\  |      |
 \_\____/
```

```
Butterfly
.-.
\  |.-.
 \_|_/
  `
```

```
Butterfly
(\o/)
(/|\)
```

```
    _* *_
  (_\|/_)
   (/|\)

a butterfly!
```

```
Butterfly
     "
   _   _
( \ | / )
  (/|\)
```

Buggy Bugs

```
    BUG>     BUG>     BUG>
  BUG      BUG      BUG
BUG      BUG      BUG        BUGBUG
  BUG      BUG      BUG    BUG    BUG>
   BUGBUGBUGBUGBUGBUGBUG
BUGBUGBUGBUGBUGBUGBUGBUGBUG>
BUGBUGBUGI<3BUGBUGBUGBUG
BUGBUGBUGBUGBUGBUGBUGBUGBUG>
   BUGBUGBUGBUGBUGBUGBUG
  BUG     BUG       BUG    BUG    BUG>
BUG      BUG      BUG        BUGBUG
  BUG     BUG       BUG
    BUG>     BUG>     BUG>
```

Caption: […] and a crawling bug

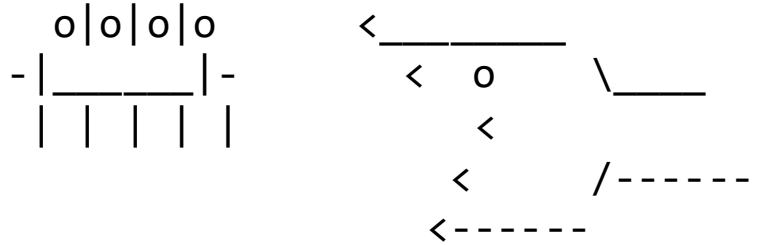
```
                                    BUGA       BU           BUGA
                                      GABUGA    BUGA          BUGA
        BUGABUGABUABUGABUGA          BUGABUGA BUGA    BUGABU
              BUGABUGABUGABUGABUGABUGABUGABUGABUGA
                        BUGA     BUGABUGABUGABUGABUGABUGABUGA
                    BUGABUGABUGABUGABUGABUGABUGABUGABUGABUGA
                        BUGA     ABUGABUGABUGABUGABUGABUGABUGA
              BUGABUGABUGABUGABUGABUGABUGABUGABUGABUGA
        BUGABUGABUABUGABUGA          BUGABUGA BUGA    BUGABU
                                      GABUGA    BUGA          BUGA
                                    BUGA       BU           BUGA
```

Caption: This is a 3 horned beatle. The body is made up of the words "BUGA", and their are 3 legs on each side of the beatle.

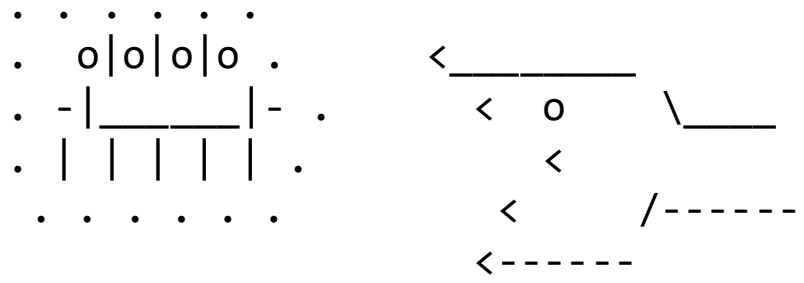
```
~~~~~ Mega Bug ~~~~~
 Bug   Bug.  Bug
BugBugBugBugBug
 Bug   Bug.  Bug
```

Story Time!

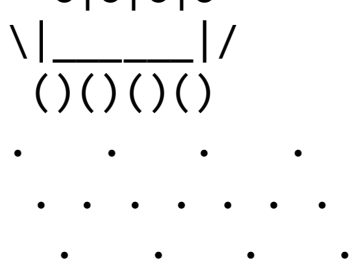
A four eyed alien appears. But a snake wants to eat it.



But wait! The ants are protecting the alien!



The snake is rebuffed and the alien is saved!It celebrates with the ants:



Lisan Al-Ghaib???

```
Watch out! I spot a Shai-Hulud (the sandworm from Dune)
~~~~~~~~~~~~~~~~~~~~
O                   O
~~~~~~~~~~~~~~~~~~~~
How did it get here all the way from Arrakis?

Phew, I think this is a regular worm
 ____________
|            :)
 ------------
How cute!

Caption: ASCII art figures of […], a sandworm from the movie
franchise "Dune", and a regular worm.
```

On Accessibility… (1/3)

Great engagement with the C0 reflection! Some themes:

- It was inspiring (and eye-opening!) to learn how blind programmers code
  - "I had never considered how a blind person would code."
  - "I found the reading of the screen window fascinating and quite unbelievable."
  - "Hearing the screen reader talk at that fast rate is very challenging and overwhelming for most people to understand."

- Small changes can make a big difference!
  - "[…] the speaker focused on the little changes that can make visual code more accessible to a wider audience. Also, his call to action to other computer scientists to make these changes as it can make platforms and apps for accessible to a wider audience.

On Accessibility… (2/3)

- Accessibility matters because…
  - "As our world is becoming 'digital-first,' ensuring software accessibility is no longer optional."
  - "[…] technology should be usable by everyone, regardless of ability. If tools and software aren't designed with accessibility in mind, they exclude talented people and limit innovation.."
  - "Accessibility is not only important for the users, to make sure everybody has access to the technology despite what their physical ability may be, but also for those developing the software, to make sure we don't have "one kind of person" working to create the resources that will be used by all. "

On Accessibility… (3/3)

Some also pointed out broader interpretations of accessibility:

“[…] accessibility in computer science can be viewed through a gender scope. Growing up, I never got into coding or computer science because as a girl, I had been socially conditioned to subconsciously believe it just wasn't for me. Now that I'm older, I know that women are directly addressing this accessibility gap through programs like "Girls Who Code" to provide an educational platform for groups that have been traditionally excluded from computer science.”

Is C0 Accessible?

Probably not…or at least **not *yet***.

When *looking* at ASCII art:

- “hearing an array of backslashes, underscores, and parenthesis won't make a clear design”
- "You can't really gather much about the art from 'drawing of bumblebee, and mosquito’."
- "Even with a caption, the creative part of the assignment is still built around a sighted interpretation. A blind developer could technically write the characters but they would not have equal access to understanding or appreciating the output."

Some ideas for improvement:

- Additional audio “signals” for certain parts of the project
- Making a tactile version of the ASCII art (e.g., braille)
- AI?

So, What?

Broadly speaking: the **digital world is inaccessible**

- but, that's changing!
- and **we** have the power to change it!

In CSE 121, we don't have the full knowledge yet to make accessible ASCII art (or Java programs, applications, video games, websites, …)

However, we encourage you to:

- think about accessibility when you make things with computers
- keep on learning more! UW is a **global leader** in digital accessibility
- e.g. at UW: CSE 443E: Accessibility, EDSPE 304, CREATE, HuskyADAPT

Agenda (3/4)

- Announcements, Reminders
- C0 Reflection Recap
- **Casting, More Variables and Operators! (now)**
- Strings and Characters Review
  - code example!

PCM: Variables

- Recall: Variables allow us to give a name to a specific value
  - 3 parts: declaration, initialization, usage
  - Example:

```
String bestBoy = "gumball";
System.out.println(bestBoy);
```

- Declaration: `int x;`
- Initialization: `x = 30;`
- Or all in one line: `int x = 30;`

PCM: Casting

- Java will do some type conversions for us
  - E.g., `int` to `double`, `double` to `String`, `int` to `String`

- ***BUT*** some conversions Java won't do for us…
  - Nonsensical conversions (e.g., `"Gumball"` to `int`)
  - Conversions that are **"lossy"** (e.g., `double` to `int`)
    - We can ask Java to **typecast** for us

```
double x = 8.83;
int xInt = (int) x;
```

New: Manipulating Variables

They're made to be manipulated, modified, and re-used!

```
int myFavoriteNumber = 7;
int tripleFavNum = myFavoriteNumber * 3;
myFavoriteNumber = myFavoriteNumber + 3;
```

Note! This doesn't really make any mathematical sense. That's because in Java, = is *assignment*, not equality!

New Operator: +=

```
myFavoriteNumber = myFavoriteNumber + 3;
```

This pattern is so common, we have a shorthand for it!

```
myFavoriteNumber += 3;
```

This works for both numeric addition and string concatenation!

More Shorthand Operators

The shorthands `-=`, `*=`, `/=`, and `%=` exist too!

```
myFavoriteNumber /= 3;
```

Should this work for integers? Doubles? Strings?

Even Shorter Shorthands

There are even shorter operators for “incrementing” and “decrementing”!

```
myFavoriteNumber++; // myFavoriteNumber += 1;
myFavoriteNumber--; // myFavoriteNumber -= 1;
```

Should this work for integers? Doubles? Strings?

Practice: Think

sli.do #cse121

What values do a, b, and c hold after this code is executed?

```
int a = 10;
int b = 30;
int c = a + b;
c -= 10;
a = b + 5;
b /= 2;
```

A. 10, 30, 40

B. 35, 15, 30

C. 35, 15.5, 30

D. 20, 15, 30

Practice: Pair

sli.do #cse121

What values do a, b, and c hold after this code is executed?

```
int a = 10;
int b = 30;
int c = a + b;
c -= 10;
a = b + 5;
b /= 2;
```

A. 10, 30, 40

B. 35, 15, 30

C. 35, 15.5, 30

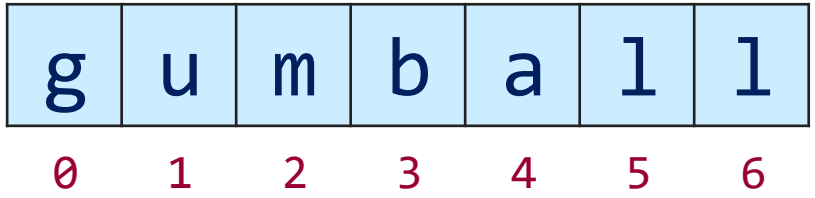
D. 20, 15, 30

Agenda (4/4)

- Announcements, Reminders
- C0 Reflection Recap
- Typecasting
- Casting, More Variables and Operators!
- **Strings and Characters Review (now)**
  - code example!

PCM: Strings & chars

- Recall: String literals are a sequence of characters that are *strung* together, begin and end with ""
  - Use zero-based indexing
- A char represents a single character
  - Begin and end with single quotes (')
  - Strings are made up of chars!




```
char letter = 'g';
char anotherLetter = 'b';
```

PCM: String Methods

Usage: `<string_variable>.<method>(…)`

Method	Description
`length()`	Returns the length of the string.
`charAt(i)`	Returns the character at index *i* of the string
`indexOf(s)`	Returns the index of the first occurrence of *s* in the string; returns -1 if *s* doesn't appear in the string
`substring(i, j)` or `substring(i)`	Returns the characters in this string from *i* (inclusive) to *j* (exclusive); if *j* is omitted, goes until the end of the string
`contains(s)`	Returns whether or not the string contains *s*
`equals(s)`	Returns whether or not the string is equal to *s* (case-sensitive)
`equalsIgnoreCase(s)`	Returns whether or not the string is equal to *s* ignoring case
`toUpperCase()`	Returns an uppercase version of the string
`toLowerCase()`	Returns a lowercase version of the string