Go Language

- Fast
- Easy to Learn
- Strongly, statically typed
- Compiled
  - int, byte, string,
  - struct,
  - built-in types for list and map
- Interfaces
- Standard library
- Testing support
- Garbage Collection
Go Language

- Fast
- Easy to Learn
- Strongly, statically typed
- Compiled
  - int, byte, string,
  - struct,
  - built-in types for list and map
- Interfaces
- Standard library
- Testing support
- Garbage Collection

Activity 8:
Guided Programming and Tutorial at https://tour.golang.org
Composite Types
Arrays

- One of the first composite data types and still used very often.
- Fixed-size collection of values, all of the same type, accessed by one or more indices:
  - sometimes called “subscripts” in analogy to mathematical notation such as $x_0, x_1, x_2, ...$ or $a_{0,0}, a_{0,1}, ..., a_{5,4}, a_{5,5}$. 

The program `array1.cpp` (in the repo):
Arrays

- One of the first composite data types and still used very often.
- Fixed-size collection of values, all of the same type, accessed by one or more indices:
  - sometimes called “subscripts” in analogy to mathematical notation such as \( x_0, x_1, x_2, \ldots \) or \( a_{0,0}, a_{0,1}, \ldots, a_{5,4}, a_{5,5} \).

**The program `array1.cpp` (in the repo):**
- two different ways to represent arrays in C++: the “traditional” C array and the “array class” in C++
Arrays

- **Index ranges**: often zero-based for ease of calculation.
Arrays

- **Index ranges**: often zero-based for ease of calculation.
- Some languages allow other ranges (see Fortran 95 program `array1.f95` in the repo.)
Arrays

- Usually stored in consecutive memory locations.

Example:

```c
int a[10];
a[0] = 99;
a[1] = -12;
a[2] = 42;
```

one int = 4 bytes, so 40 bytes.
Arrays

- Usually stored in consecutive memory locations.
- Example:

  ```
  int a[10];
a[0] = 99;
a[1] = -12;
a[2] = 42;
  ```

  one int = 4 bytes, so 40 bytes.
Arrays

To find an int array element, take its base location plus its index times 4:

- $a[2]$ is in location:
  $$10000 + 2 \times (\text{sizeof int}) = 10000 + 2 \times 4 = 10008$$
What about two-dimensional arrays?

```c
char c[3][4]
```
What about two-dimensional arrays?

```c
char c[3][4]
```

one char = 1 byte, so 12 bytes.
What about two-dimensional arrays?

```c
char c[3][4]
```

one char = 1 byte, so 12 bytes.

**We have a choice!**
Arrays

ROW MAJOR ORDER

- char c[3][4]
  - c[1][2] = base address
  - + (1*(# cols) + 2)*sizeof char
  - =10000+(1*4+2)*1=10006
Arrays

COLUMN MAJOR ORDER

- char c[3][4]
  
  c[1][2] = base address
  + (2*(#rows)+1)*sizeof char
  = 10000+(6+1)*1 = 10007
Java, C, and many other languages use row major order. (See program array2.cpp in the repo.)
Java, C, and many other languages use row major order. (See program array2.cpp in the repo.)

Fortran uses column major order. (See program array2.for in the repo.)
Arrays

- Java, C, and many other languages use row major order. (See program `array2.cpp` in the repo.)
- Fortran uses column major order. (See program `array2.for` in the repo.)
- Does it matter?
Efficiency Concerns

In a language that uses row-major order, it is more efficient to access elements by rows because they are in consecutive locations; accessing by columns involves a lot of jumping around in memory.
Efficiency Concerns

In a language that uses row-major order, it is more efficient to access elements by rows because they are in consecutive locations; accessing by columns involves a lot of jumping around in memory.

See programs byrows.c and bycols.c; byrows.for and bycols.for; ByCols.java and ByRows.java in the class repo.

TODO:
1. Run experiments - each program 3 runs, record your time results.
2. Consult with your group on the results of your experiments.
3. Record your observations in the Google Form given through Slack.
If arrays are just consecutive locations in memory, how does the computer know whether, e.g., an array is 3 by 4 or 4 by 3 or 2 by 6 or ...?
Other Aspects of Arrays - Bounds

- If arrays are just consecutive locations in memory, how does the computer know whether, e.g., an array is 3 by 4 or 4 by 3 or 2 by 6 or ...?
- It needs to remember this information somewhere – symbol table (static arrays) or “dope vector” (dynamic arrays) – see page 364.
Dope Vector

Descriptor (Dope Vector)

- VO: Virtual Origin
- LB 1 (= 1): Lower Bound on Subscript 1
- UB 1 (= 3): Upper Bound on Subscript 1
- LB 2 (= -1): Lower Bound on Subscript Z
- UB 2 (= 1): Upper Bound on Subscript Z
- E (= 1): Size of Component
Dynamic Arrays

- In C, array size is known at compile time unless we allocate memory from the heap and use pointers to access it.

Try program array3.c in the repo for an example using "malloc".
Dynamic Arrays

- In C, array size is known at compile time unless we allocate memory from the heap and use pointers to access it.
- Try program `array3.c` in the repo for an example using “malloc”.

Consider the C or Java declaration:

```java
int a[3][4];
```

- This can be considered as an “array of arrays”:
- an array `a[3]` whose elements are of type “int [4]”.
- Once we allow this, we can allow more general “arrays of arrays”, e.g., `Array.java`
“Set” - like classes in Java; e.g., program `Sets.java` shows the `HashSet` class.
Sets

- "Set" - like classes in Java; e.g., program Sets.java shows the HashSet class.
- Some languages have a built-in set data type.
  Python:
  ```python
  >>> s = {1,2,4,5,1,2,3,5,3,2,1}
  >>> s
  set([1, 2, 3, 4, 5])
  ```
Since C is often used by systems programmers, it is convenient to be able to directly access and manipulate memory addresses.

However, care must be taken. How many of you have ever seen the words?
segmentation fault