* Composite/Aggregate Types
  - Records/Structures
  - Arrays/Lists
  - 2-D Arrays
  - strings

* A4 is due on Thurs 11/7
* Exam 2 is on Tue 11/12

11/5

Pre-Defined Types

- Numbers
  - Integers
  - Floats
  - Complex
- Characters
- Booleans
- Enumerated Types
- Subrange Types

Composite/Aggregate Types

- Records/Structures
- Arrays
- Strings
- Sets
- Hash Tables
- Lists
- files
- Images
- etc.

Q.1: How are they defined?

Q.2: How are they used?

Records/Structures

e.g. Place – city, state, zip, population

Bryn Mawr, PA, 19010, 5879
string string string int

PASCAL: records

how

```
type place = record
       city : string;
       state : string;
       zip : string;
       population: integer;
     end;
```

use

```
var bm : place;
bm = ("Bryn Mawr", "PA", "19010", "5879");
```

(city, state, zip, population)

this is positional association.

```
bm.city
bm.state
 etc -
```

C → structures/structs

(1)

how

```
struct place {
    char *city;
    char *state;
    char *zip;
    int population;
};
```

use

```
struct place bm;
bm = { "Bryn Mawr", --};
bm.city - - -
```

(2)

```
typedef struct {
    |
    ≡
    |
} place;
place bm;
```

Python: tuple (a, b, c)

How: bm = ("Bryn Mawr", "PA", "19010", 5879)

use: bm[0] ← indexed like lists
bm[i]

Also,
city, state, zip, population = bm

Java - use class

```
class Place {
    private string city;
    private string state;
    private string zip;
    private int population;

    public Place (String c, String s, String z, String p){
        city = c;
        population = p;
    }
}
//class Place
```

How

use: Place bm = new Place ("Bryn Mawr", "PA", ... , 5879);

if fields are public	if private
bm.city	bm.getCity()
	bm.getPopulation()
	- etc -

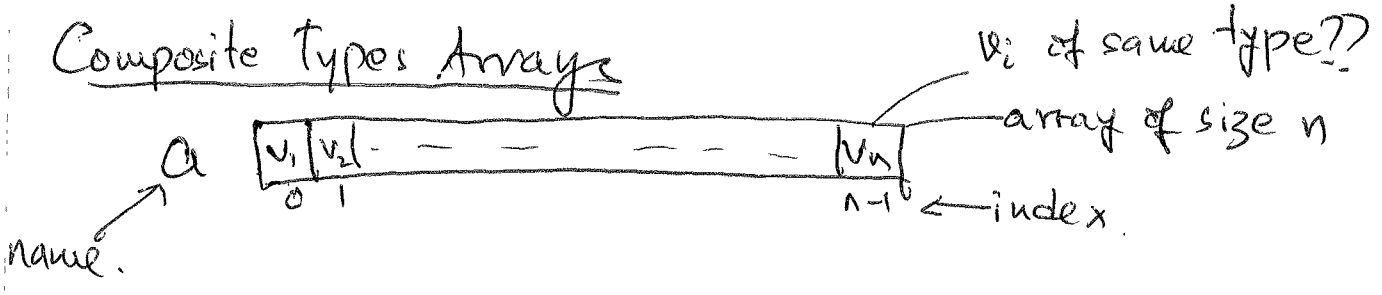
Theory

if $T_1, T_2, T_3 \ldots$ are types

A record/struct/tuple is a cross-product (a, b, c)

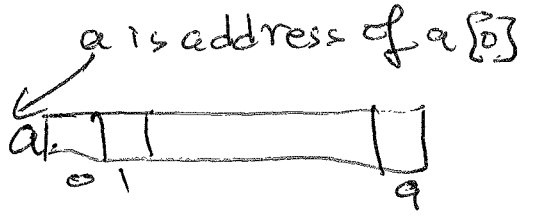
where $a \in T_1$
$b \in T_2$
$c \in T_3$
etc.

Composite Types: Arrays



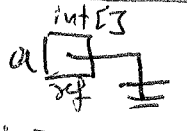
Declaration

```
int a[10]; //C/C++
int[] a;   // Java
```



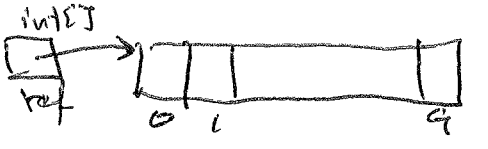
Declaration vs Creation/Construction

```
int[] a;
```

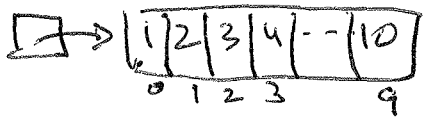


JAVA

```
int[] a = new int[10];
```




```
int[] a = {1,2,3,4,5,6,7,8,9,10};
```



Also in C

```
int a[] = {1,2, ---10};
```

Indexing: a[i]

Arrays in Python? → LISTS

a = [1, 2, 3, 4, ..., 10]

a[0]
a[1]

In Python, lists can have diff. types of elements

slicing is allowed
a[3:7]
a[:3]
a[3:]
a[:]

See lab 4 for more details

Memory Allocation

where and when memory for arrays is allocated depends on whether the PL provides means for declaration separate from construction + where in a program the array is defined.

e.g. C

```
#include <stdio.h>

int main() {
    int a[10];
}//main
```

→ Stack frame of main()

```
#include <stdio.h>

int a[10];
int main() {
}//main.
```

→ Static storage area

Also depends on whether a value or reference model is used.

JAVA (reference model) `int[] a` a → [ ]

`a = new int[10];` a → [ ] (0 … 9)

allocated on the heap

Important Bindings for arrays

1. Name ⟶ a
2. Type of elements in a ⟶ int
3. size ⟶ 10 (#elements in array)
4. Index bounds ⟶ [0..n-1]

Homogeneous Composite Arrays: often all elements are the same type.

2-Dimensional Arrays

$$A = \begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0n} \\ a_{10} & a_{11} & \cdots & a_{1n} \\ \vdots & \vdots & & \vdots \\ a_{m0} & a_{m1} & \cdots & a_{mn} \end{bmatrix}_{m \times n}$$

C/C++ `int A[3][4];` ⟶ creates a 3×4 matrix/array

Java `int[][] A = new int[3][4];`

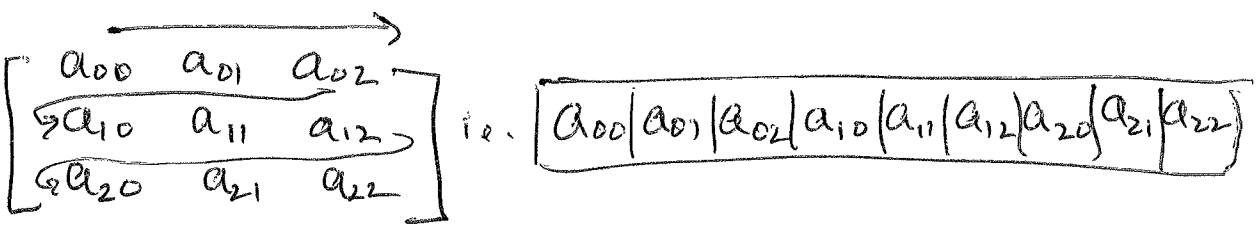
Indexing `A[i][j]`

Python $A = [\,[a_{00}, a_{01}, a_{02}], [a_{10}, a_{11}, a_{12}], [a_{20}, a_{21}, a_{22}]\,]$

⟶ `A[i][j]`

Storing 2-D Arrays

Row-major Form



Elements are stored sequentially, one row after the other.

Column-Major Form

Elements are stored sequentially one column after the other:

$a_{00}|a_{10}|a_{20}|a_{01}|a_{11}|a_{21}|a_{02}|a_{12}|a_{22}$

Both C + Java store 2-D arrays in row-major form.

Implication: when iterating over a 2-D array it is more efficient to traverse rows-first.

i.e.

Preferred

```
for (int r=0; r<M; r++) {   // rows
    for(int c=0; c<N; c++) {
        |=
    }
}
```

OR

Not Preferred

```
for (int c=0; c<N; c++) {   // columns
    for (int r=0; r<M; r++) {
        |
    }
}
```

How is addrss of a[i][j] computed?

Row-Major

a[i][j] = base addrss + W(i*N + j)

where base address is addrss of first element (a[0][0])
W is word size in bytes (e.g int W=4)
N is the number of columns

Column-Major

a[i][j] = base addrss + W(j*M + i)

M is the number of columns

Alternative Multi-Dimensional Array Representations

- Sparse Matrices
- Iliffe Vectors - Java, Swift, use these.

Composite Types - Strings

C, C++, Java, Python

"Hello" "Hello\n World"

in Python, strings can be enclosed in single, double, or triple quotes.

"Hello", 'Hello', """Hello""", '''"Hello"'''

In C: Strings have to be null terminated

Strings are arrays of char

Java has String type

Python: str