Enumeration Types - Enable you to define your own type name and literals.

Example: Pascal

```pascal
type weekday = (Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday);
var day: weekday;
```
Pascal provides two operations on enumeration types:

- `succ(Monday)` -> Tuesday
- `pred(Monday)` -> Wednesday

We can write loops:

```pascal
for today := Sunday to Saturday do ...
```

Array Indexes:

```pascal
var attendance : array[weekday] of integer;
... attendance[Sunday] = 0;
```

C also has enumerated types:

```c
enum weekday {Sunday, Monday, ..., Saturday};
weekday day = Sunday;
for (int i = Sunday; i <= Saturday; i++) {...}
// enums are like ints!
```

Java (enumerated types are equivalent to a class)

```java
enum Weekday {Sunday, Monday, ..., Saturday};
Weekday day = Weekday.Sunday;
for (Weekday d : Weekday.values) {...}
```

Neither Python, nor Go have enumerated types.

Bigger question: Are enumerated types really useful??
Subranges:

Pascal

```pascal
type workday : Monday..Friday; (* Base Type: weekday *)
    score : 0..100; (* Base Type: integer *)
```

Ada also has them. Mostly missing in modern programming languages.

*Same Question: While they have some use, they are not considered so useful as to be included in modern programming languages.*
13 Composite/Aggregates types - Records

Records/Structures

e.g. Place Pi;

city, state, zip, population

int a [10];

Pascal - record

type student = record
  firstname : string;
  lastname : string;
  year : integer;
  major : string;
end;

use
var hp : student;


pre-defined/ Built-in Types (scalars)

- Numbers
  - Integers
  - Floating point
  - complex
- Characters
- Booleans
- Enumerated Types
- Subrange types

Composite/Aggregate Types

- Records/structures
- Arrays
- Strings
- sets
- Hash tables
- Lists
- files

13 Composite/Aggregates types - Structures

struct student {
  char *first;
  char *last;
  int year;
  char *major;
}

Use

struct student hp;

hp = %"Harry", %", 1998, %";

hp.major

C++
Go

type student struct {
    first string
    last string
    year int
    major string
}

use

    var hp student hp.year

hp = student{
    first: "Harry"
    last: "Potter"
    year: 1998
    major: "Wizardry"
}

also


Python - Tuple

    (a, b, c)

    [hp[0] indexed like arrays

Java

    class student {
        private String first, last, major;
        int year;

        public student( )
        {
            first = --;
        }

    Student hp = new Student("", ",", 1998, ",");
Arrays

Construction mechanism:

\[
\begin{align*}
\text{C:} & \quad \text{int} \, a[10] \quad \text{// C} \\
\text{Java:} & \quad \text{int}[10] \ a = \text{new int}[10]; \quad \text{// Java}
\end{align*}
\]

Use:
- \( a[i] \) indexing
- \( a[i:j] \) slicing

All elements in an array are of the same type (homogenous composite type)

1. Name - a
2. Type - int
3. Size - 10 ints
4. Index bounds - [0..9]