Last Lecture

1) Static Variables

allocation
deallocation

Today

2. Stack-Dynamic Variables

Binding is created when the decleration statement is elaborated.

Stack Dynamic

```
void xyz(int b)
Stack-Dynamic ⟶ int x;
      :
      :
}
```

```
void main(){
      :
   xyz(5);
}
```

Stack dynamic variables are created on the dynamic.

3. Explicit heap-dynamic Variables

– Alloboration / declaration are done through special instructions.

Ex: C++

```
int * node;
node = new int;
   :
   :
delete node;
```

Special Explicit Instructions

Ex c

```
int * list;
list = malloc( 20 * sizeof(int));
list[5] = 48;
   ⋮
free(list);
```

Special Instructions

Ex Java

```
 Integer node;
 Node = new Integer(8);
     ⋮
 node = null;
```

Special Instructions

Heap-Dynamic variables are often used to construct dynamic data structures like linked list, trees, etc.

4. Implicit Heap-Dynamic Variables

Variables are assigned storage when they are assigned values.

Ex Javascript

```
hiphs = [ 74, 84, 86]; // An array of integers is created on heap and bound
                       // to the variables highs.
```

Disadvantage: Array subscripts change every time a new assignment happens.

Type Checking

The activity of ensuring operands of an operator compatible types.
Otherwise a type error is reported

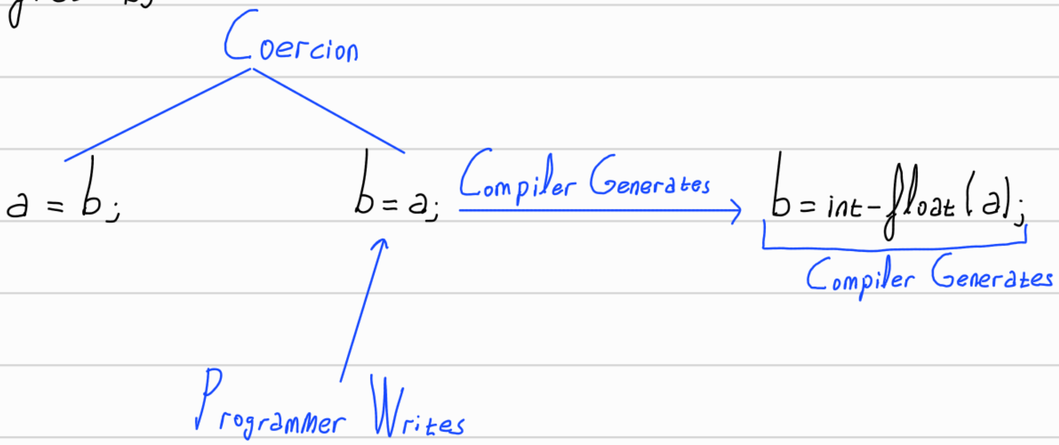
Ex

```
a * b;  // If a and b have numeric types, no type error is
        // Otherwise a type error.
```

Coercion: Automatic Type Conversion

Compiler adds a function

```
int   a;
float b;
```



Coercion

a = b;

b = a; $\xrightarrow{\text{Compiler Generates}}$ $\underbrace{b = \text{int-float}(a);}_{\text{Compiler Generates}}$

Programmer Writes

- Casting: Programmer defined conversion

```
a = (int) b;
```

In Java

```
String a;
int    b;

b = (int) a; // Compiler Error
```

Static Type Checking

- Checking type compability at compilation

Dynamic Type Checking

- Checking compatibility at runtime,

Strong Typing

- A language design criteria
- A language is strongly-typed if type errors are always detected.
- Type errors can be caught at compiletime or runtime.
- Ada is a strongly-typed language (also C# and Java are)

Ex Too see why strong typing is important

```
int a, b;
float d;
```

Programmer meant to write a+b , but mistakenly typed a+d; If the language is strongly typed it gives a type error.