Module 2: Data Types & Variables – RCBScript

1. Data Types

- **number** → auto type detection (int or float)
- **text** → Unicode string
- **bool** → `yes` or `no`
- **char** → single character
- **handle<Type>** → managed reference (instead of raw pointers)

---

2. Variable Declaration & Initialization Rules

Syntax:

```
varName:type = value
```

Examples:

```
 a:number = 10
b:number = 3.14
c:text = "Hello World"
d:bool = yes
```

```
e:char = 'X'
f = make handle<number>
f = 25
```

- Rules:
    - Type **must be declared** explicitly.
    - If initialized, type must match the assigned value.
    - Handles must first be created with `make handle<Type>`.
    - Re-assignment to mismatched type is a **Type Error**.

---

3. Error Message Specification

Format:

```
Line <n>: <Error Type> → <description>
```

Examples:

- `Line 3: Type Error → expected number, got text`
- `Line 5: Reference Error → discard called on undeclared handle`

```
- Line 7: Initialization Error → variable used before assignment
```

---

4. Example Programs

✅ Correct Usage

```
a:number = 42
b:number = 3.14
c:text = "Hello RCBScript"
d:bool = yes
e:char = 'Z'

ref = make handle<number>
ref = 99

results("a = " + a)
results("c = " + c)
```

❌ Incorrect Usage (error case)

```
a:number = 10
a = "text"    # ERROR: expected number, got text
```

Compiler Output:

```
Line 2: Type Error → expected number, got text
```

---

5. Testing Framework Integration

```
func sum(a:number, b:number) -> number
    reply a + b

test "sum of integers"
    expect sum(2,3) is 5

test "variable type test"
    x:number = 5
    y:text = "hello"
    expect x is 5
    expect y is "hello"
```

---

6. Work Division

👤 Bhukya Lachiram Nayak

- Wrote **documentation for Data Types & Variables**.

- Defined **rules for declaration & initialization**.
- Drafted **error message formats + examples**.
- Created **error demonstration programs**.
- **Files**:
    - `data_types.rcb`
    - `error_examples.rcb`

👤 Vineeth

- Finalized **type system semantics** (number auto-detects int/float, handle<Type>).
- Built **example programs for declarations**.
- Added **test cases for type validation**.
- **Files**:
    - `type_system.rcb`
    - `variable_examples.rcb`

❖ SPECIFICATION DOCUMENT FOR 8 WEEKS

Week 1: Core Language Specification Setup

- **Tasks:**
    - Define philosophy, design goals, and guiding principles.
    - Finalize **keywords list** & **basic syntax rules** (statements, blocks, indentation).
    - Decide comment style (`#` and `###`).

- **Deliverables:**
    - Language Overview Document (like you wrote).
    - Draft BNF grammar for basic expressions.
- **Responsibility:**
    - **Person A:** Grammar + keywords.
    - **Person B:** Documentation + overview formatting.

Week 2: Data Types & Variables

- **Tasks:**
    - Finalize **primitive data types** (`number`, `text`, `bool`, `char`).
    - Define rules for **handle<Type>** memory references.
    - Document variable declaration & initialization rules.
- **Deliverables:**
    - Data type specification doc.
    - Example programs for variable declaration.
- **Responsibility:**
    - **Person A:** Type system + examples.
    - **Person B:** Documentation + error messages for type errors (e.g., “Type mismatch: expected number, got text”).

Week 3: Functions & Control Flow

- **Tasks:**
    - Define **function declaration syntax** (`func`) and return typing.
    - Specify **control flow constructs**: `if`, `else`, `loop from`, `loop while`, `loop until`.
    - Add **match (pattern matching)** draft design.
- **Deliverables:**
    - Control flow specification doc.
    - Example test programs (loops, conditionals, recursion).
- **Responsibility:**
    - **Person A:** Function syntax + semantics.
    - **Person B:** Control flow + test examples.

Week 4: Object-Oriented Features

- **Tasks:**
    - Define **class declaration syntax** (`class`, `public`, `private`).
    - Specify **methods** inside classes.
    - Document **object creation** with `make` and destruction with `discard`.
    - Introduce **record** for immutable structures.
- **Deliverables:**
    - OOP specification doc.
    - Example programs for Person, Hello classes.
- **Responsibility:**

- **Person A:** Class & record syntax.
- **Person B:** Memory handling (`make`, `discard`) + error messages (e.g., “discard called on undeclared handle”).

Week 5: Standard Library (Phase 1)

- **Tasks:**
  - Design **basic I/O library** (like `print`, `input`).
  - Add **basic math functions** (`sum`, `pow`, `sqrt`).
  - Draft **string utilities** (concatenation, length).
- **Deliverables:**
  - Library specification doc.
  - Example test cases using library.
- **Responsibility:**
  - **Person A:** I/O library design.
  - **Person B:** Math + string utilities.

Week 6: Error Handling & Testing Framework

- **Tasks:**
  - Define **error reporting style**:
    - e.g., `Line 3: Syntax Error → missing semicolon` (but adapt since semicolons aren’t used).
    - e.g., `Line 5: Type Error → expected number, got text`.

    - Specify built-in **test framework** (`test "addition" ... expect ...`).
    - Write initial suite of **unit test cases**.
- **Deliverables:**
    - Error message format guide.
    - Test framework specification doc.
- **Responsibility:**
    - **Person A:** Error reporting style & formats.
    - **Person B:** Test framework design + examples.

Week 7: Advanced Features & Polishing

- **Tasks:**
    - Refine **pattern matching (`match`)**.
    - Document **scope rules & visibility (public/private)** clearly.
    - Add future roadmap (planned lists/arrays).
- **Deliverables:**
    - Extended specification doc (with advanced features).
    - Example programs using `match`.
- **Responsibility:**
    - **Person A:** Scope rules + visibility.
    - **Person B:** Pattern matching + examples.

Week 8: Integration, Review & Final Documentation

- **Tasks:**
    - Integrate all specs into one **Language Specification Document**.
    - Finalize **library reference guide** (with examples).
    - Collect all **test cases & sample programs** into a suite.
    - Prepare **final presentation/demo** of the language design.
- **Deliverables:**
    - Full Language Spec (PDF/Doc).
    - Library Guide.
    - Test Suite.
    - Example Programs (Hello, calculator, loops, OOP).
- **Responsibility:**
    - **Person A:** Final Spec Document.
    - **Person B:** Library + Test Suite.