!This project is also available in the repo: https://github.com/usman-s-mahmood/compiler-construction-project-cosmos!

Compiler Construction – Project Phase 1

COSMOS Programming Language

A Mini C++-like Language Inspired by Astrophysics & Dr. Neil deGrasse Tyson

Submitted by: M. Usman Shahid

Roll No: L1F22BSCS1057

Section: G-10

Submitted to: M/s Aneela Mahmood

Date: November 17, 2025

---

1. Language Concept – Why “COSMOS”?

I watched Dr. Neil deGrasse Tyson’s famous TV series **“Cosmos: A Spacetime Odyssey”** on Discovery, and it completely changed how I see the universe. He explains black holes, stars, light-years, and the Big Bang in such a simple, beautiful, and human way that even a normal person can fall in love with science.

That inspired me to create **COSMOS** – a programming language where every keyword is a real astronomical term. Instead of boring words like `int`, `return`, `+`, the language uses `star`, `supernova`, `fusion`. The code reads like a poetic story about the universe while being technically correct. My goal was to prove that programming languages can be both functional and inspiring.

---

2. My 16 Original Keywords (All Astrophysics-Themed)

Keyword	Real Science Meaning	Meaning in COSMOS Language
universe	The entire cosmos	main function
star	Luminous ball of plasma	integer (int)
planet	Body orbiting a star	float / double
galaxy	Collection of billions of stars	string
nebula	Cloud of gas and dust	function declaration
observe	To look at the sky	print / cout
orbit	Path around a celestial body	if
gravity	Universal force	else
supernova	Explosive death of a massive star	return
blackhole	Point of no return	break
quasar	Extremely bright active galaxy	continue
lightyear	Distance light travels in 1 year	while loop
eventhorizon	Boundary of a black hole	for loop (reserved)
darkmatter	Invisible matter	const
astrophysics	Study of the universe	class (reserved)
cosmic	Relating to the cosmos	void

3. My Original Operators & Punctuations

Token	Lexeme	Meaning in COSMOS
launch	launch	Assignment (=)
fusion	fusion	Addition (+)
collapse	collapse	Subtraction (−)
radiate	radiate	Multiplication (×)
expand	expand	Division (÷)
::	::	Scope resolution
{ } ( ) ;	Standard C++ punctuations used exactly as in C++	

4. Regular Expressions Table

Token Type	Regular Expression (Flex)	Example Lexemes
Identifier	`[A-Za-z_][A-Za-z0-9_]*`	`age`,`distance`,`Proxima4`
Integer	`[0-9]+`	`13800000000`,`42`
Float	`[0-9]*\.[0-9]+([eE][+-]?[0-9]+)?`	`3.14`,`0.000004848`
Scientific Notation	`[0-9]+[eE][+-]?[0-9]+`	`9.46073e15`,`1.38e10`
String	`"([^"\n]	.)*"`
Keywords	`universe|star|planet|galaxy|nebula|observe|...`	`observe`,`supernova`,`orbit`
Word Operators	`fusion|collapse|radiate|expand`	`fusion`,`radiate`
Assignment	`launch`	`launch`
Punctuation	`{	}
Invalid Identifier	`@.*`	`@salary` → ERROR
Any other character	`.`	`#`,`$` → ERROR

5. Error Handling

- Identifiers starting with `@` are reported as invalid (common mistake in local coding).
- Any unrecognized character is printed as:

  **Line X: ERROR → **

Example:

```
Line 7: ERROR → @salary (invalid identifier)
Line 12: ERROR → #
```

6. Sample Program (22 lines) – `cosmos_demo.cosmos`

universe main() {

```
star age;

planet distance;
```

```
  galaxy message;

  observe "Calculating cosmic distance...";

  distance launch 9.46073e15;    // 1 light-year in meters

  age launch 13800000000;       // age of universe

  distance launch distance radiate 3;

  distance launch distance fusion 5.8786e12;

  observe "Age of Universe: " age;

  observe "Distance traveled: " distance;

  orbit (age expand 10000000000) {

  observe "We are in the Stelliferous Era!";

  } gravity {

  observe "Entering Black Hole Era...";

  }

  supernova 0;
```

}

The scanner produces a `tokens.txt` with correct line numbers and token types.
