Google Summer of Code

python SOFTWARE FOUNDATION

# Implementation of Strings functionalities, interactivity, and parser benchmark


Name: Abdelrahman Khaled Fouad

University: Mansoura university

Github: Abdelrahman-Kh-Fouad

Gitlab: Abdelrahman-kh-fouad

LinkedIn: Abdelrahman-Kh-Fouad

Time Zone: Cairo, Egypt (UTC +2)

## About me:

- I'm a Computer Science student at the Faculty of Computers and Information, Mansoura University - Egypt (fourth year).
- I have completed the basic courses in Computer Science (Algorithms and data-structure, Computer Architecture, Networks, etc.) this semester; also, I have a compiler design course.
- I am interested in competitive programming, practicing problem-solving, and participating in competitions.

## Background and experience:

- I'm coding in C++, python, and a little java; mainly, I'm using Linux (Kubuntu).
- My experience in C++ was in several projects i made by SMFL and problem-solving.
- I have some experience in web development from my 2021 internship, which made me familiar with flask apps and vue.js, and a little of Kubernetes.

## My projects:

- Contained C++:
  - First two projects of CS106x stanford (programming abstraction in C++) using (C++ ,SMFL):
    - Fauxtoshop
    - GameOfLife
  - Some of my solutions of problems(different judges)
- Contained python:
  - Vjudge-Solution: get all solutions on vjudge and organize them.
  - Movies: pypt app to search for movies rate and add some movies to favorites.
  - INI-Parser: parser and manipulator for ini files.
  - TODO: todo website is written in vue.js and flask app.

- Others:
  - Pac-man clone: I implemented a maze generator and shortest path for enemies(Java, Android SDK).

## Open source experience:

I have experience in open-source contributions.

- Pydatastructs: I implemented 2d BIT and generalized BIT(Binary Indexed Tree) in #495 (Not merged yet).

## Availability and time zone:

In the first three weeks of the code period, I will be in my exams to contribute work in other weeks. I will be working only on the project during the summer, communicating with mentors regularly, and being available full-time (35 hours per week).

My time zone is (UTC +2 Cairo).

## Contributions to Lpython:

- Merged :
  - Add max() , min() #335 : implement built_in function max and min to get maximum of two, three elements or minimum of two, three elements.
  - Change doctest.h to newer version(Ondřej Čertík idea) #309
- Opened:
  - –time-report implementation #359: time report option in lpython to gave a time report in each stage.
  - Parser benchmark #394: benchmark script to run timers for multiple files.

## Motivation:

i'm always interest in programming language and some concepts of computational theory, so lpython give me a big opportunity to contribute in python compiler, this powerful language, and to take adventure to learning compilers in depth.

## Project Idea:

The project idea is to add two features from ASR and LLVM levels and benchmark for the new parser.

- Implementing strings functionality: this feature is mainly in asr and llvm levels; the main idea here is to add string functionalities like:
  - Concatenating.
  - Passing to and from functions.
  - Lists of strings.
  - Any function that mentors needs for other features.

  (functionalities mentioned by the mentor: Ondřej Čertík )
  So I will learn llvm ir for (asr to llvm stage) and learn about (ast to asr stage), and I already have a little experience with ast to asr stage from implementing **max() and min()** functions.

  Recently mentors have decided to implement intrinsic functions with a new approach, and it will make it easy to handle problems of strings functions
  So I will learn llvm, and if there is anything related to strings while implementing this new approach, I will make it.

  After I learn llvm I will add the main functionalities that the mentors discussed and decided to implement during the Bonding period.

- Adding interactive option: interactive option is the mood of the compiler that evaluates line by line in an interactive way with the user.
  Like every python interpreter like ipython, interactive is very useful for users; it makes it easy for a programmer to see results in an immediate moment, and it is used by data scientists allows to see results periodically, so it's very important to implement in lpython.

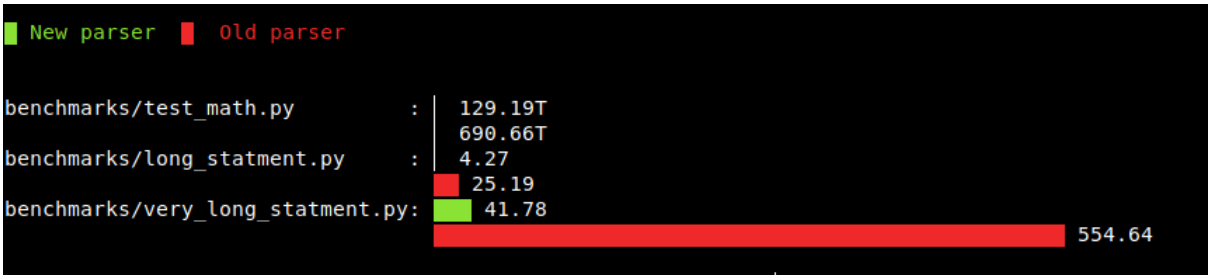
  Interactivity implemented already in LFortran, and we already have infrastructure from Lfortran and utilities like tpl dir that contain terminal

functionalities and some functions that print results in the terminal; also lpython evaluator is ready, I will add anything missing in the evaluator and implement the whole interactive mood **in src/bin/lpython.cpp .**

- Make parser benchmark: there is a new parser now that is being implemented, and we want to estimate its speed in comparison to the older parser(python parser).

  The first thing I must implement is a timer in the parsing stage in lpython to estimate the time that takes a new parser; in Lfortran there is an option **–time-report** that shows time results of every stage (reading-file, parsing, ast to asr, and asr to llvm), so I should make this time report also in Lpython, and with this, we can use this report to know parser time, and users could see this report if they choose **–time-report** option.

  And I have already implemented it on this pull request.
  The next step is making a benchmark itself, and it will compare files and estimate their parsing time; the workflow here will be the same as the tests workflow, we will have a dir we put on it files we want to compare, and it will run them all and compare times, and the output will be results graph like this:




  And the basic implementation for benchmark in this pr.

## Time period :

- **Community Bonding Period** (May 20, 2022 - June 12, 2022)
  - Get to understand the codebase. Specifically, llvm ir part.
  - Discuss the ideas, their difficulties, and ways to solve them with mentors and their recommendations for suitable strings functionalities to be implemented and finalize the time plan.

- I'll start coding in the middle of this period if the time is more than required.

- **Coding phase** (June 13, 2022 - September 04, 2022)

  - Week 1 - 3 (June 13 - July 3 )
    - In this period, the final exams of the spring will start. This is not official since the semester agenda is not yet posted. In this period, it'll not be very easy to actively contribute to the project. Nonetheless, I'll be present on the zulip discussing the project and keeping up with the updates.

  - Week 4 - 6 (July 4 - July 24)
    - I continue learning llvm and starting to implement some string functionalities.
    - I will try to fix any functions doesn't complete because of strings functionalities.
    - Finish basic benchmark parser with basic output graph.

  - Week 7 - 9 (July 25 - August 14)
    - I will finish strings functionalities that I discuss with mentors during the Bonding period
    - I will finish the benchmark and make generalize for a lot of options for comparison if the project needs this in the future.

  - Week 10 -12(August 15 - September 4)
    - I will bring the infrastructure and code we will use for interactivity from Fortran.
    - I will add necessary features to the evaluation methods

  - Week 13 (September 5 - September 11)
    - I will be finalizing my work and adding all things in docs or (make comments and articles to describe my work and help people in the future).

  If I have a chance to start coding during the bonding period, I will start To make up for the exams period.

If time permits or if I've finished earlier, the rest of the time should be spent solving issues and continuing the parser benchmark if it is not completed yet.

## Post Gsoc:

I like the project, and I will continue to contribute anything to it; and if there is a regular opportunity to contribute regularly in lpython, Fortran compilers, I will apply on it.
And if these projects will be in gsoc next year will apply to be a contributor.