Compiler Construction Assignment One

Jawad Ahmed(20P-0165) Section: BCS-6A

February 17, 2023

1 Evolution of Programming languages

- **Mention the form of input data, such a punching cards**
- **Also mention the level of language such as high level language, low level language or machine level language**
- **If a language is designed for a particular task such as web-designing, research, engineering, write the name of the language with the associated area of field**

Ans: Programming languages is a fundamental unit of today tech world. We can make machines to solve our problems with the help programming languages. Let's talk about some history of programming languages. In the computer worls there are 500+ programming languages having their own syntax and features. The father of the computer is Charles Babbage but the interesting thing is that Charles Babbage has not written the first code. It was Ada Lovelace who has written first ever programming language. The first code was written for computing Bernoulli's number.

At the start computers are made they are big as room. In that we insert cards to tell machine to do any calculations these are called punching cards. Let's understand punching cards with this example In assembly we have an add function that add two numbers. There is a specific code of add when you assemble it that will be converted to machine code. To that equivalent code we can create a card that card has slots some slots you will be going to punch and some slots you left. You can think as binary the punch slots represent 1's and other represent 0's. So as in assembly you write 'add al, 5' after assemble that code to machine code some binary made find that binary and write that in the puching cards so this is your assembly add instruction on the punching cards. The difference is that now instead of binary in some memory these instruction is on the card and all of your program is written on the cards. So there will be an operator who will be going to take cards and arrange them that in what sequence I will going to give these cards to the machine the get the required result. That is the mechanical system and you can implement the fetch decode cycle in that also. As more inventions are made these mechanical systems become very diffcult to manage so we have shifted towards Electronics.

First there are machine language there are all 0's and 1's. That is very difficult to do programming in the machine language. Assembly language was created as an exact shorthand for machine level coding, so that you wouldn't have to count 0s and 1s all day. It works the same as machine level code with instructions and operands. The first assembly language, "Contracted Notation", was developed in 1947 by the late Kathleen Booth. The language doesn't look anything like "modern" assembly though it's more a mathematical representation of computer operations. It was eventually realized that programming in assembly language required a great deal of intellectual effort.

The first functioning programming languages designed to communicate instructions to a computer were written in the early 1950s. John Mauchly's Short Code, proposed in 1949, was one of the first high-level languages ever developed for an electronic computer.[6] Unlike machine code, Short Code statements represented mathematical expressions in understandable form. However, the program had to be interpreted into machine code every time it ran, making the process much slower than running the equivalent machine code.

The period from the late 1960s to the late 1970s brought a major flowering of programming languages. Most of the major language paradigms now in use were invented in this period. C, an early

systems programming language, was developed by Dennis Ritchie and Ken Thompson at Bell Labs between 1969 and 1973. In 1960s and 1970s there is a debate over the mertis of 'Structured programming' which means that programming without the use of 'goto'. A significant fraction of programmers believed that, even in languages that provide "goto", it is bad programming style to use it except in rare circumstances. This debate was closely related to language design: some languages did not include a "goto" at all, which forced structured programming on the programmer.

The 1980s were years of relative consolidation in imperative languages. The imperative programming means a software development paradigm where functions are implicitly coded in every step required to solve a problem. C++ combined object-oriented and systems programming.

The rapid growth of the Internet in the mid-1990s was the next major historic event in programming languages. In particular, the JavaScript programming language rose to popularity because of its early integration with the Netscape Navigator web browser. Various programming languages achieved widespread used in developing customized applications for web servers such as PHP. Many "rapid application development" (RAD) languages emerged, which usually came with an IDE, garbage collection, and were descendants of older languages. All such languages were object-oriented. These included Object Pascal, Objective Caml, Visual Basic, and Java. Java in particular received much attention.

Currently Programming language evolution continues, in both industry and research. Some of the recent trends have included:

- Construct to support concurrent and distributed programming.
- Pure functional programming for making code easier to reason about and easier to parallelize.
- Increased interest in distribution and mobility.
- Massively parallel languages for GPU graphics processing units and supercomputer arrays, including OpenCL.
- Early research into quantum computing programming languages
- Early research in applying Artificial Intelligence techniques to generate code using AI like using GPT-3

There are different languages designed for different tasks. For example for the web designing there are several languages that work together to create visually appealing and functional websites. That include HTML, CSS, Javascript, PHP, SQl, Bootstrap, Jquery, Python, Ruby and much more.

For the research and engineering there are multiple langauges designed like Python(has large number of libraries that provide support for scientific computing and data analysis), MATLAB (desinged for numerical computing and scientific visualization), R(For statistical computing and used in data analysis), Julia and so on. It depends upon your problem you are going to solve and see the features of different langauges and then decide which language will be more suitable for solving your problem and gives you optimzed solution.

2 Generations of programming languages

- **Mention details of all generation (from 1st generation of languages (1GL up to 5GL))**
- **Mention few names of programming language in each generation**
- **Effect of Generations of Programming languages on the performance of computer**

Ans: The first two generations of programming languages is considered as 'Low Level Languages' and the third, fourth and fifth generation are considerent 'High Level languages'.

First-Generation Language: The first generation language is also called the machine language. This language is machine dependent that mean the program written in one machine will not work on the other machine. The machine language statements are written in binary code (0/1 form). These programs are very fast as they are written in binary and they need no translation and they will directly run. The disadvantage for these languages is that for human it is very difficult to write programs in binary and if any error occured there will be very difficult to debug that. Due to these reason new languages were introduced to make programming easy. The examples of first generation programming languages are all the languages that are written in binary in 1's and 0's forms.

Second-Generation Language: The second generation programming languages include assembly languages. Assembly language is human readable as compared to machine language. Assembly languages converted to machine code with the help of assembler. Assembly is easy as compared to machine language and errors are easy to solve. The assembly language is machine-dependent. The second generation programming languages include **Autocode, FORTRAN II and assembly language**.

Third-Generation Language: The third generation programming languages are also called procedural lanaguages. These language are easy to understand as it consists of the use of a series of English like words that human can understand easily and that's why they are considered as 'High Level Languages'. These languages are translated to machine code with the help of Compiler/Interpreter. The third genration programming languages are **BASIC, COBOL, Pascal, Fortran, C, C++, Perl and Ada**.

Fourth-Generation Language: Fourth generation programming languages are high level languages that are closer to human and non-procedural language. It enables user to access the database. These languages has memory consumption, Less flexible and has poor control over hardware. The examples of fourth generation languages are **SQL, MATLAB, Mathematica, R, Perl, Python, Rubly** and so on.

Fifth-Generation Language: The fifth generation programming languages are based on artificial intelligence. These languages use visual tool to develop a program. Parallel Processing and superconductors are used for this type of language to make real artificial intelligence. These languages have complex and long code and they require alot of resources and they are really expensive. These languages are typically used in areas such as artificial intelligence, expert systems, and natural language processing. The examples of fifth generaton programming languages are **PROLOG, LISP, OPS5, Lisp, Haskell** and much more.

3 What are scripting languages? Elaborate with an example

Scripting language is a language that does not require compilation and is instead interpreted one line at a time during runtime. They are frequently used to create dynamic web applications nowadays. The two types of scripting languages are server-side scripting languages and client-side scripting languages. Server side scripting languages include Python, PHP and Perl and client side scripting languages include Javascript and perl. The objective of these languages to frequently communicate with other programming languages.

Example: Let's say you have a website that you want to be online only from 11:00pm to 5:00am. Now to host that website you will be using some virtual machine. First solution for the website availability is that every day at exact 11:00 pm you visit you access your virtual machine and make your website online. The second and best solution will be is to automate this task. We can use bash scripting that will start the nodejs server (to make website online) at 11:00pm and stop the server at 5:00 am. The script for this is really simple as shown below. There are plenty of tasks for which scripting languages used that is one I have discussed to show the power of scripting languages and what are the scenarios scripting languages will be helpful.

```
#!/bin/bash

# Start the Node.js server at 11pm
```

```
if [ $(date +%H) -eq 23 ]; then
  echo "Starting Node.js server at $(date +%Y-%m-%d\ %H:%M:%S)"
  node /Desktop/my-app/server.js &
fi

# Stop the Node.js server at 5am
if [ $(date +%H) -eq 5 ]; then
  echo "Stopping Node.js server at $(date +%Y-%m-%d\ %H:%M:%S)"
  pkill node
fi
```