EasyChair Preprint

№ 6682

A Research: Data Structures in Leelus Programming Language

Frank Appiah

September 25, 2021

A Research:

Data Structures in Leelus Programming Language.

Frank Appiah[1]

ABSTRACT. This is a research on data structures using pointer declaration for array dimensional type of representation in variable declaration. In this research, I will describe type array, term array and value array drawn from both data structure and type system.

KEYWORDS. Data Structure, Type System, Leelus, Programming Language, Type Array, Term Array, Value Array.

1 Introduction

Leelus C++ is a typed language theory for interpretable artificial intelligence method for multi-agent perspective. This is based on Object Oriented Language. In describing, readers will be looking at type constructs that includes type, term and value in array structures. In programming languages, a type system is a logical system comprising a set of rules that assigns a property called a type to the various constructs of a computer program, such as variables, expressions, functions or modules.[1] These types formalise and enforce the otherwise implicit categories the programmer uses for algebraic data types, data structures, or other components (e.g. "string", "array of float", "function returning boolean").

In Leelus type language, each data type is declared as a character pointer to represent an array of characters. The type definition are as follows in Leelus:

typedef char* action;
typedef char* interest;
typedef char* temporal;
typedef char* location;

[1] King's College London, Department of Informatics, London, England, UK.

The first programmed action declared as acts[0] is an equalisation of "buy" action. A buy action still defines an action of buying process in terms of business or economics. It can describe buy types like goods, services or both.

The second programmed action is termed as acts[1] to represent a value of sell action. Sell value is an action of economic type. So are all the other two types.

typedef char* rank;

I will now program an action and then structure into an array pointer dimension:

action a="buy", a1="sell", a2="sold", a3="trade";

Data Structure:

action* acts;

acts[0]= "buy" ; acts[2]= "sold";

acts[1]= "sell" ; acts[3]= "trade";

Data System can be a structure of a tree:

Blue : Type

Green: Term

Sea Blue: Value

Turquoise: Array Structure

You say that action* is an action array or action pointer.

Action array has four terminal values including:

- Buy
- Sell
- Sold
- Trade

Action array is a data structure.

The type system is in C++ declaration. I will now show type theory to study the system. We will be looking at type, term and values in tabular form. I will also describe the typing environment, and denotation of judgement.

On Data Structure Details:

Data Structure	Type Array	Term Array	Value Array
action* acts;	action	acts	Buy

Data Structure	Type Array	Term Array	Value Array
	action	acts	Sell
	action	acts	Sold
	action	acts	Trade

Typing **action** type system gives an environment listed as a pair:

- *acts[0] : action*
- *acts[1]: action*
- *acts[2]: action*
- *acts[3]: action*

These listed pairs are **assignments** in the action type system:

- *acts[0]: buy*
- *acts[1]: sell*
- *acts[2]: sold*
- *acts[3]: trade*

The denotation of judgements are as listed:

- Γ |- *acts[0]: buy*
- Γ |- *acts[1]: sell*
- Γ |- *acts[2]: sold*
- Γ |- *acts[3]: trade*

The next set of programmed interests are described in terms of their value in the program. The code for declaration is as follows:

interest* ints;

ints[o]= “sell_pc”;

ints[1]= “buy_pc”;

On Data Structure Details:

Data Structure	Type Array	Term Array	Value Array
interest* ints;	Interest	ints	sell_pc
	Interest	Ints	buy_pc

Two values of interest are used in the program : sell_pc or buy_pc. Interest is a business term and a calculation on the product of principal, time and rate in percentage.

Typing **interest** type system gives an *environment* listed as a pair:

- *ints[0]: interest*
- *ints[1]: interest*

These listed pairs are *assignments* in the **interest** type system:

- *ints[0]: sell_pc*
- *ints[1]: buy_pc*

The denotation of **judgements** are as listed:

- Γ |- *ints[0]: sell_pc*
- Γ |- *ints[1]: sell_pc*

Interest array is a type structure of array type.

Businesses are ranked in indexes to stay relevant in the market. The store value of this relevancy is of rank type. Always or sometimes delivery of quality of service is an interest of business in playing agent parties. The declaration of rank is coded as follows:

rank* rans;

rans[0]=“rank_9”;

rans[1]=“rank_7”;

On Data Structure Details:

Data Structure	Type Array	Term Array	Value Array
rank* rans;	rank	rans[0]	rank_9

Data Structure	Type Array	Term Array	Value Array
	rank	rans[1]	rank_7

Higher rank has low quality of service than a lower rank value.

Typing rank type system gives an **environment** listed as a pair:

- rans[0]: rank.
- rans[1]: rank.

Assignments of the **rank** system are:

- rans[0]: rank_9
- rans[1]: rank_7

The denotation of judgements are as listed:

- Γ |- *rans[0]: rank_9*
- Γ |- *rans[0]: rank_7*

The next set of programmed locations are described in terms of their value in the program. The code for declaration is as follows:

location* locs;

locs[0]="tottenham";

locs[1]="London";

On Data Structure Details:

Data Structure	Type	Term	Value
location* locs;	location	locs[0]	tottenham
	location	locs[1]	London

This points to location of business situated at both Tottenham and London. But other local businesses can be located elsewhere.

Typing location type system gives an **environment** listed as a pair:

- *locs[0]:location*

- *locs[1]:location*

Next is the **assignment** of location system:

- *locs[0]:tottenham*
- *locs[1]:London*

On denotation of **judgement**:

- Γ |- *locs[0]: tottenham*
- Γ |- *locs[1]: London*

Doing business cannot be a permanent activity. Holding to do business with an agent in an agency is temporary in some sense. After a certain period of time limits an organisation in doing business with the agent company.

The next data structure of programmed temporals are described in terms of their value in the program. The code for declaration is as follows:

temporal* temps;

temps[0]="Sat 21/11/2021 12:00pm";

temps[1]="Tues 01/09/2022";

Typing temporal type system gives an **environment** listed as a pair:

- *temps[0]: temporal*
- *temps[1]: temporal*

Next is the **assignment** of temporal system:

- *temps[0]:Sat 21/11/2021 12:00pm*
- *temps[1]:Tues 01/09/2022*

Temporal System has a **denotation of judgement**:

- Γ |- *temps[0]: Sat 21/11/2021 12:00pm*

- Γ | - *temps[1]: Tues 01/09/2022*

2 CONCLUSION

This is about data structures in Leelus programming language using pointer of array dimensions. In here, I use both data structure and type system concepts to describe type array, term array and value array. I did show tabular forms of these descriptions. Type system concepts like denotation of judgement, assignment and environment of typing are used in this article.

Further Reading

[1] Frank Appiah(2021). A Type Theory of Leelus Type System in C++ Programming Language. ScienceOpen Preprint. DOI: 10.14293/S2199-1006.1.SOR-.PPZQLTI.v1

[2] Josuttis, N. M. (2012). The C++ standard library: a tutorial and reference.

[3] Coplien, J. O. (1991). Advanced C++ programming styles and idioms. Addison-Wesley Longman Publishing Co., Inc..

[4] Stroustrup, B. (2000). The C++ programming language. Pearson Education India.

[5] Horowitz, E., Sahni, S., & Rajasekaran, S. (1997). Computer algorithms C++: C++ and pseudocode versions. Macmillan.

[6] Eckel, B. (2021). Thinking in C++.

[7] Frank Appiah (2021). An Object-Oriented Language Based on C++ :Leelus Typed Language. EasyChair Preprint no. 6189.

[8] Frank Appiah (2021). The Mapping of the Logical Structure, Enact to the Engagement Structure of Enact. EasyChair Preprint no. 6187 .

[9] Frank Appiah (2021). Semantic Computation of the Propositional Model Composites in Enactment Logic. EasyChair Preprint no. 6186.