CSE 425: Concepts of Programming Languages

SLecture

Imperative Programming

Group: ioradiX

Md Abir ---------- 1621775042

Mostak khan khadem ---------- 1621777042

Nasir Ahmad ---------- 1612658642

Introduction

Imperative programming is a paradigm of computer programming in which the program describes a sequence of steps that change the state of the computer. Unlike declarative programming, which describes "what" a program should accomplish**.**

History of Imperative Languages :

- First imperative languages: assembly languages
- 1954-1955: Fortran (FormulaTranslator)
  John Backus developed for IBM 704
- Late 1950's: Algol (Algorithmic Language)
- 1958: Cobol (Common Business Oriented Language) Developed by a government committee; Grace Hopper very influential.

Imperative programming is divided into three broad categories:

- Procedural
- Object Oriented Programming(OOP)
- Parallel processing.

Procedural programming paradigm

This paradigm emphasizes on procedure in terms of under lying machine model. There is no difference in between procedural and imperative approach. It has the ability to reuse the code and it was boon at that time when it was in use because of its reusability.

Examples of Procedural programming paradigm:

Language	Developed By
C	Dennis Ritchie and Ken Thompson
C++	BjarneStroustrup
Java	James Gosling at Sun Microsystems
Pascal	Niklaus Wirth

Object oriented programming

The program is written as a collection of classes and object which are meant for communication. The smallest and basic entity is object and all kind of computation is performed on the objects only. More emphasis is on data rather procedure. It can handle almost all kind of real life problems which are today in scenario.

Language	Developed By
Visual Basic .NET	developed by Microsoft
Ruby	Yukihiro Matsumoto
Python	Guido Van Rossum

Parallel processing approach

Parallel processing is the processing of program instructions by dividing them among multiple processors. A parallel processing system possess many numbers of processor with the objective of running a program in less time by dividing them. This approach seems to be like divide and conquer. **Examples** are NESL (one of the oldest one) and C/C++ also supports because of some library function.

Imperativelanguages are Turing complete if they support integers, basic arithmetic operators, assignment, sequencing, looping and branching.

Modernimperative languages generally also include features such as

- Expressions and assignment
- Control structures (loops, decisions)
- I/O commands
- Procedures and functions
- Error and exception handling
- Library support for data structures

Data Types in Imperative

In computer science and computer programming, a data type or simply type is an attribute of data which tells the compiler or interpreter how the programmer intends to use the data. FORTRAN (John Backus 1954) was a compiled language that allowed named variables, complex expressions, subprograms, and many other features now common in imperative languages. Basically data types in imperative programming language depends on the language itself. Generally, there are two types of data type :

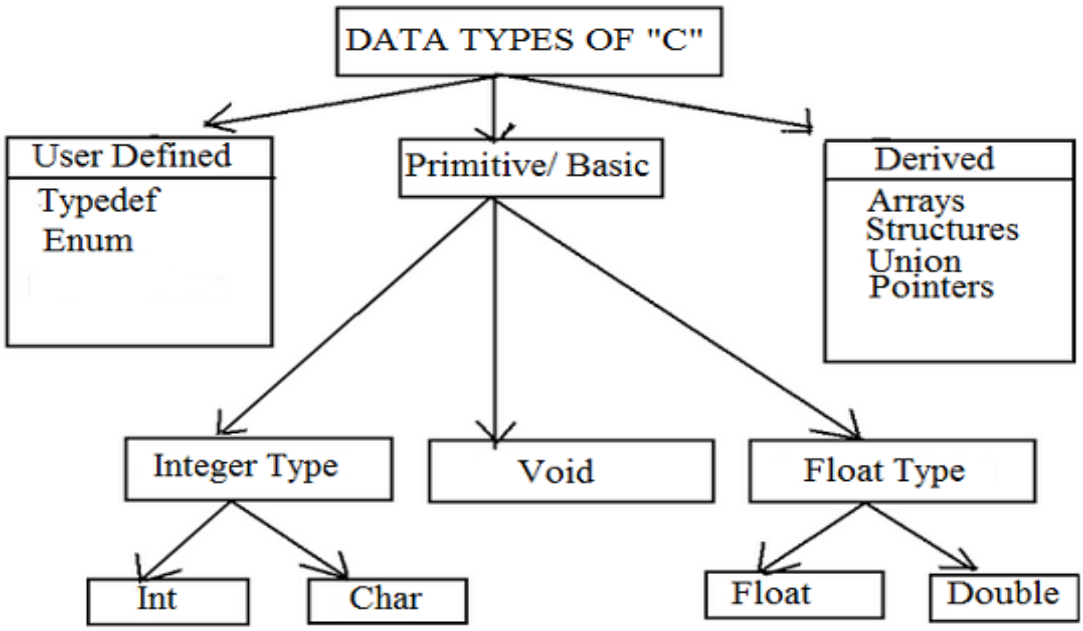
Primitive Data Type:

Primitive data types are predefined types of data, which are supported by the programming language. Programmers can use these data types when creating variables in their programs.

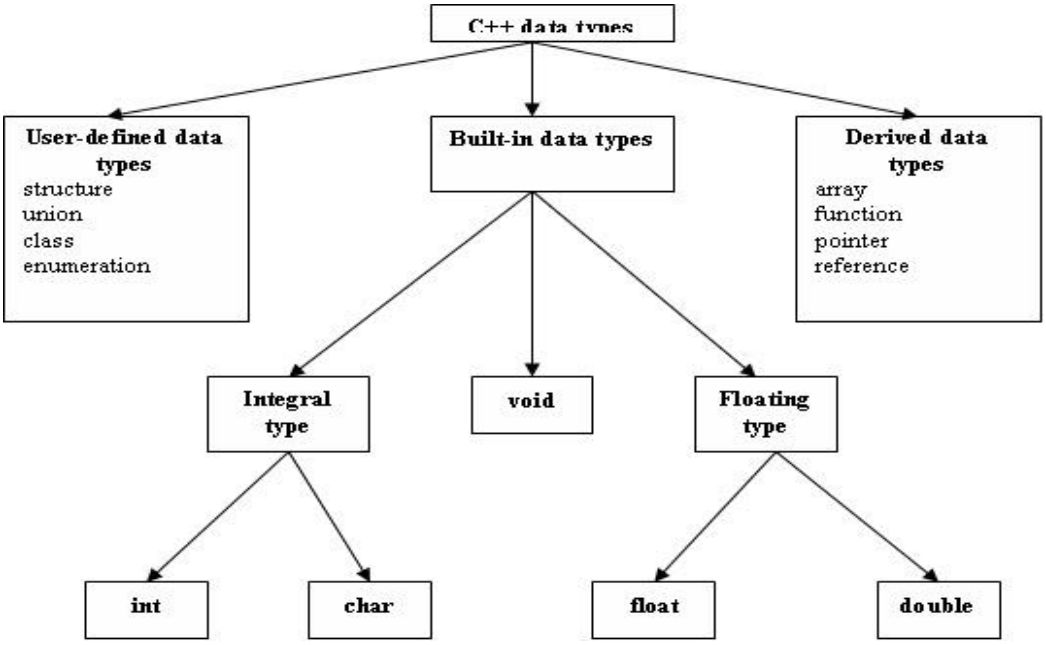
Non Primitive Data Type:

Non-primitive data types are not defined by the programming language, but are instead created by the programmer. They are sometimes called "reference variables," or "object references," since they reference a memory location, which stores the data.

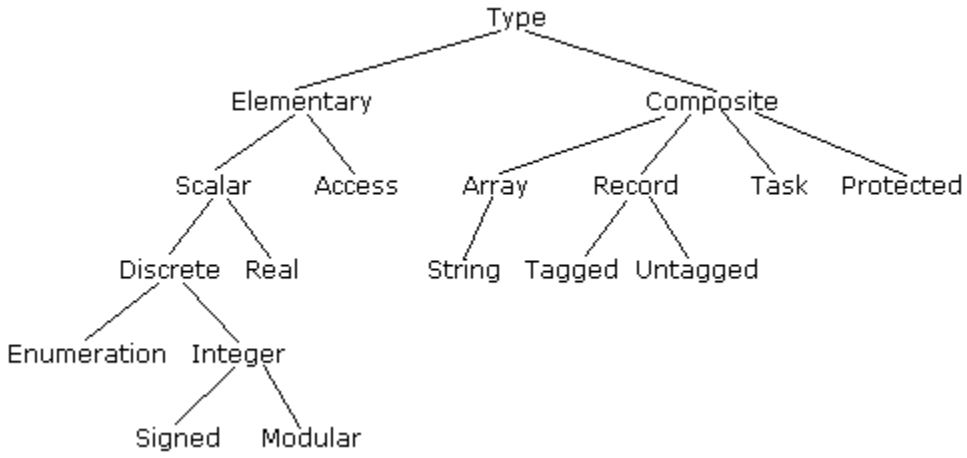
An example for Data Types in C



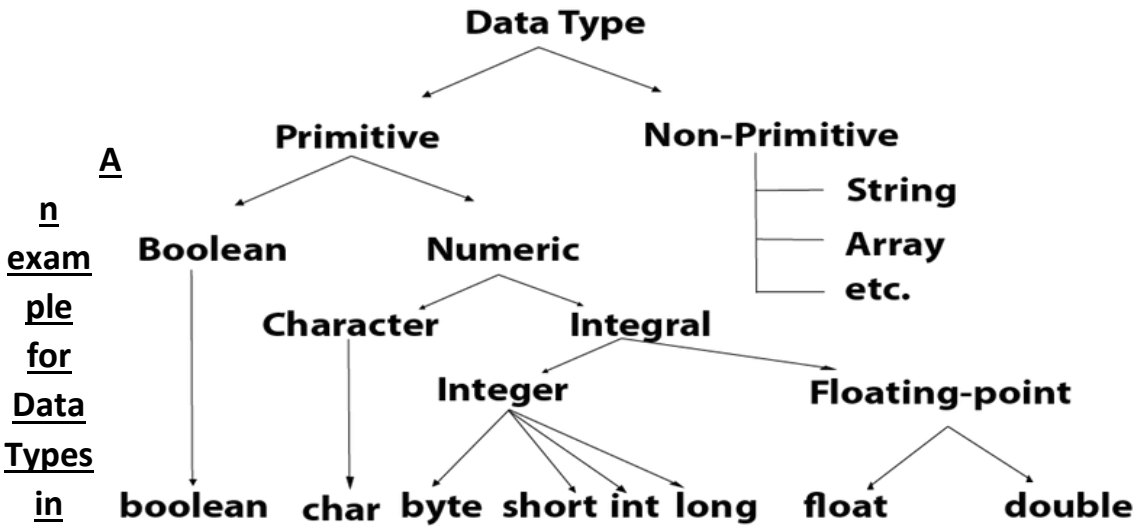
An example for Data Types in C ++



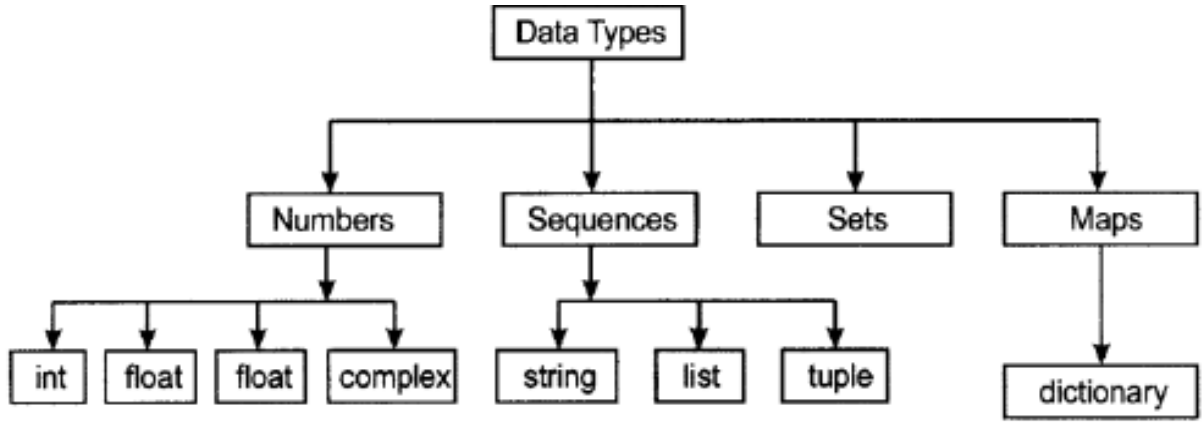
An example for Data Types in Ada



An example for Data Types in Java



An example for Data Types in python



Python Data types

Advantages & Disadvantages of Imperative Programming Language

Language	Advantages	Disadvantages
C (was originally designed for and implemented on the UNIX operating system on the DEC PDP-11, by Dennis Ritchie)	• Relatively small language • Can generate very efficient code • Runs on virtually all platforms	• Lacks of Exception Handling • C does not have the concept of OOPs • Low level of abstraction
C++ (The language was designed with the intent of merging the efficiency and conciseness of C with the object-oriented programming features of SIMULA-67)	• It is a multi-paradigm programming language :logic, structure • Gives the programmer the provision of total control over memory management • Use of Pointers	• C++ does not support any built-in threads. • Lacks the feature of a garbage collector to automatically filter out unnecessary data. • Security issues exist due to the availability of friend functions, global variables and, pointers.
Perl (was originally developed by Larry Wall in 1987 as a general-purpose Unix scripting language)	• Originally designed for text processing • Supports several paradigms: imperative, object-oriented, functional • Used for Web applications, graphics processing	• It is an interpreted language • It is too slow for large programing • The syntax of perl is non interval
Java (an object-oriented programming language developed by James Gosling and colleagues at Sun Microsystems in the early 1990s)	• Multithread supported • Platform-Independent • Supports Object-Oriented	• Java is memory-consuming and significantly slower than natively compiled languages such as C or C++ • Single-Paradigm Language
Pascal (was originally developed in 1970 by Niklaus Wirth and is	• Incorporated a variety of data types • Possible to work with	• Comparatively slower

named after the famous French mathematician Blaise Pascal)	complex data types • General purpose language	
Fortran (Originally developed by IBM[3] in the 1950s for scientific and engineering applications)	• Generates the fastest native code • Is highly optimized for victimization • Surprisingly readable and easy to understand.	• Lack of inherent parallelism • Lack of dynamic storage • Lack of numeric portability
Phthon Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991.	• Supports Object-Oriented • Extensive support Libraries • Productivity	• significantly slower than natively compiled languages such as C or C++ • Python has limitations with database access • Python is not a very good language for mobile development

String

In imperative programming language there are two criteria to design the string .The two most important design issues that are specific to character string types are the following:

- Should strings be simply a special kind of character array or a primitive type?
- Should strings have static or dynamic length?

The design criteria basically language dependent like :

C/C++ :

Strings are not defined as a primitive type, string data is usually stored in arrays of single characters. C and C++ use char arrays to store character strings. These languages provide a collection of string operations through standard libraries.

JAVA :

In java strings are supported by the String class, whose values are constant strings, and the StringBuffer class, whose values are changeable and are more like arrays of single characters. These values are specified with methods of the StringBuffer class.

- C# and Ruby include string classes that are similar to those of Java.

PYTHON :

Python includes strings as a primitive type and has operations for substring reference, catenation, indexing to access individual characters, as well as methods for searching and replacement. There is also an operation for character membership in a string. So, even though Python's strings are primitive types, for character and substring references, they act very much like arrays of characters. However, Python strings are immutable, similar to the String class objects of Java.

String Length Options

There are several design choices regarding the length of string values.

- **Static :**
  The length is set when the string is created . Such a string is called a static length string.
  Example :C#,Python,.NET,RUBY,
- **Limited Dynamic :**
  Allow strings to have varying length up to a declared and fixed maximum set by the variable's definition.
  Example : C,C++
- **Dynamic length:**
  Allow strings to have varying length with no maximum limit.
  Example : JavaScript, Perl,php

There are three approach of designing string :

Approach 1:

Strings can be stored as linked-list, when strings grows the new cell come from the heap and linked with the existing list. Here is one problem with that approach there need extra storage

Approach 2:

Use array of pointers and the problem with that approach is it requires extra storage but faster processing

Approach 3:

Typically this one is used to store complete string in a new adjacent cell and remove / deallocate the old version of the string . it is faster approach and efficient memory management

Pointers

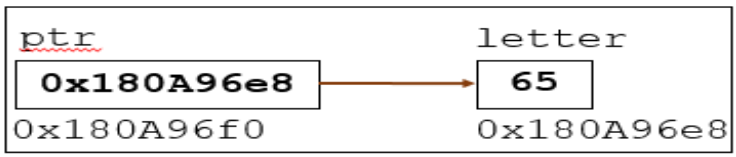
In computer science, a pointer is a programming language object that stores the memory address of another value located in computer memory. A pointer references a location in memory, and obtaining the value stored at that location is known as dereferencing the pointer.

The general form of a pointer variable declaration is –

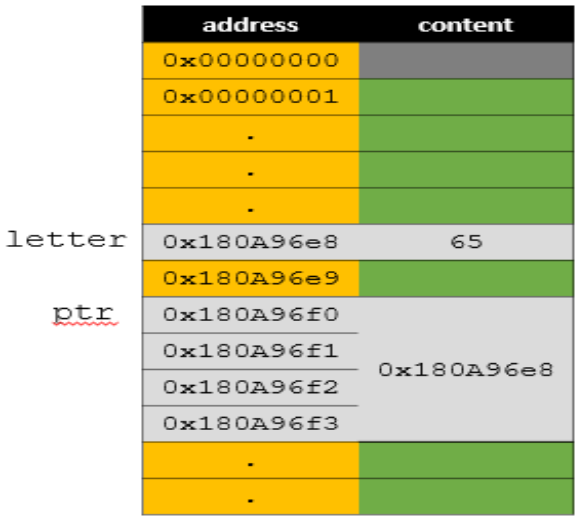
```
type *var-name;
int    *ip;   /* pointer to an integer */
double *dp;   /* pointer to a double */
```

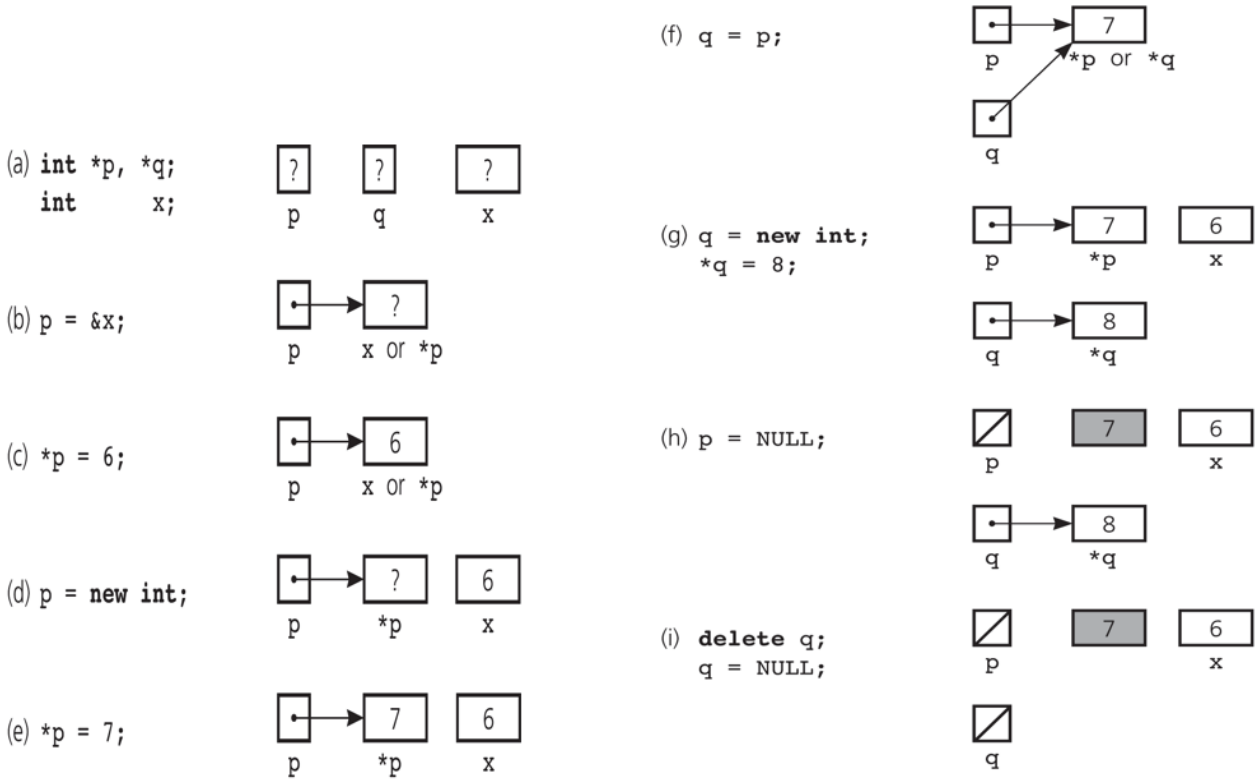
Why would we want to use pointers?

- To call a function by reference so that the data passed to the function can be changed inside the function.
- To create a dynamic data structure which can grow larger or smaller as necessary.



The variable **ptr** contains the address of **letter**





Parameter Passing by Pointer

```
#include <stdio.h>

void swap(int a, int b)int main(void)

{                                                    {

int temp;int x= 5, y= 10;

temp = a;printf("Before swap function: x = %d, y = %d\n",x,y);

a = b;                                               swap(x,y);

   b = temp;printf("After swap function: x = %d, y = %d",x,y);

}                                                    }
```

Output:

Before swap function: x = 5, y = 10

After swap function: x = 10, y = 5

Array of ponters

In computer programming, an array of pointers is an indexed set of variables in which the variables are pointers (a reference to a location in memory).

An example

```
#include <stdio.h>
constint ARRAY_SIZE = 5;
int main ()
{
  /* first, declare and set an array of five integers:    */
intarray_of_integers[] = {5, 10, 20, 40, 80};
/* next, declare an array of five pointers-to-integers: */
inti, *array_of_pointers[ARRAY_SIZE];

for ( i = 0; i< ARRAY_SIZE; i++)
  {
    /* for indices 1 through 5, set a pointer to
point to a corresponding integer:                */
array_of_pointers[i] = &array_of_integers[i];    Output
  }
array_of_integers[0] = 5
for ( i = 0; i< ARRAY_SIZE; i++)array_of_integers[1] = 10
  {array_of_integers[2] = 20
    /* print the values of the integers pointed to array_of_integers[3] = 40
by the pointers:                              */
printf("array_of_integers[%d] = %d\n", i, *array_of_pointers[i] );
  }

return 0;
```

```
        }
```

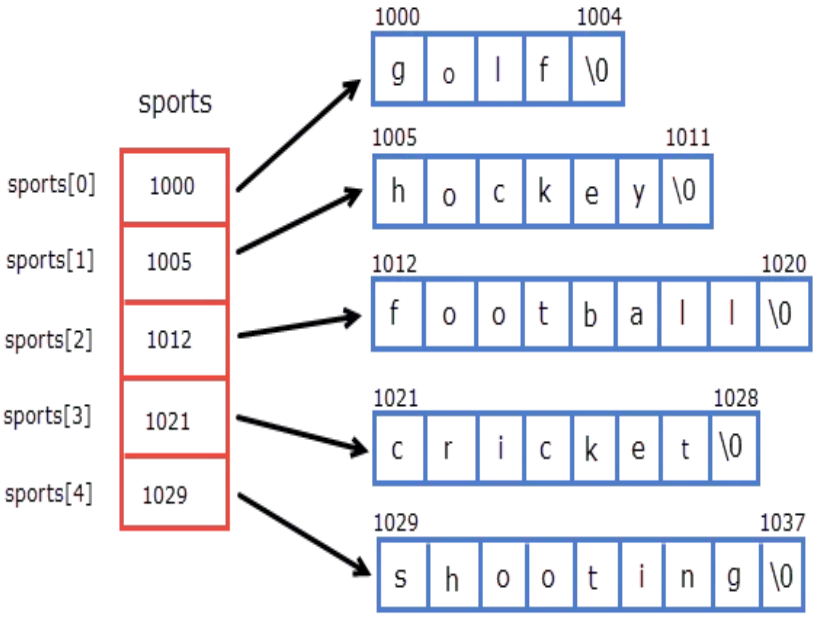
Or

```
char *sports[] = {
        "golf",
        "hockey",
        "football",
        "cricket",
        "shooting"
                };
```

It is important to note that each element of the sports array is a string literal and since a string literal points to the base address of the first character, the base type of each element of the sports array is a pointer to char or (char*).

The 0th element i.earr[0] points to the base address of string "golf". Similarly, the 1st element i.earr[1] points to the base address of string "hockey" and so on.

Here is how an array of pointers to string is stored in memory.



Memory representation of array of pointers

Reference

https://docs.scipy.org/doc/numpy-1.13.0/reference/arrays.scalars.html

https://www.javatpoint.com/data-types-in-c

https://www.sarthaks.com/131871/classify-the-python-data-types

https://www.computerhope.com/jargon/a/array-of-pointers.htm

https://overiq.com/c-programming-101/array-of-pointers-to-strings-in-c/

https://en.wikipedia.org/wiki/Wiki

https://www.tutorialspoint.com/index.htm

https://www.startertutorials.com/ppl/books/sebesta.pdf