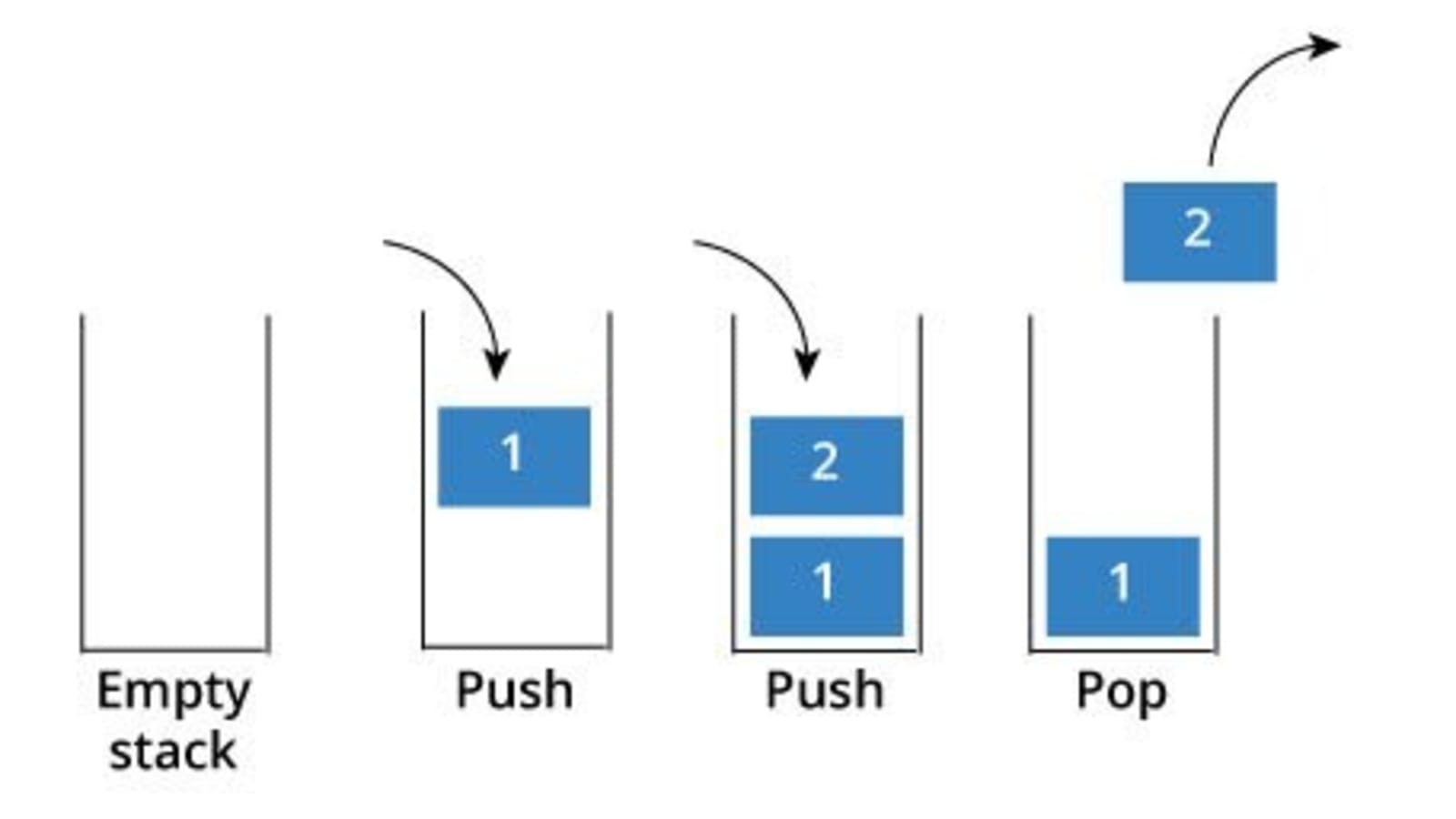
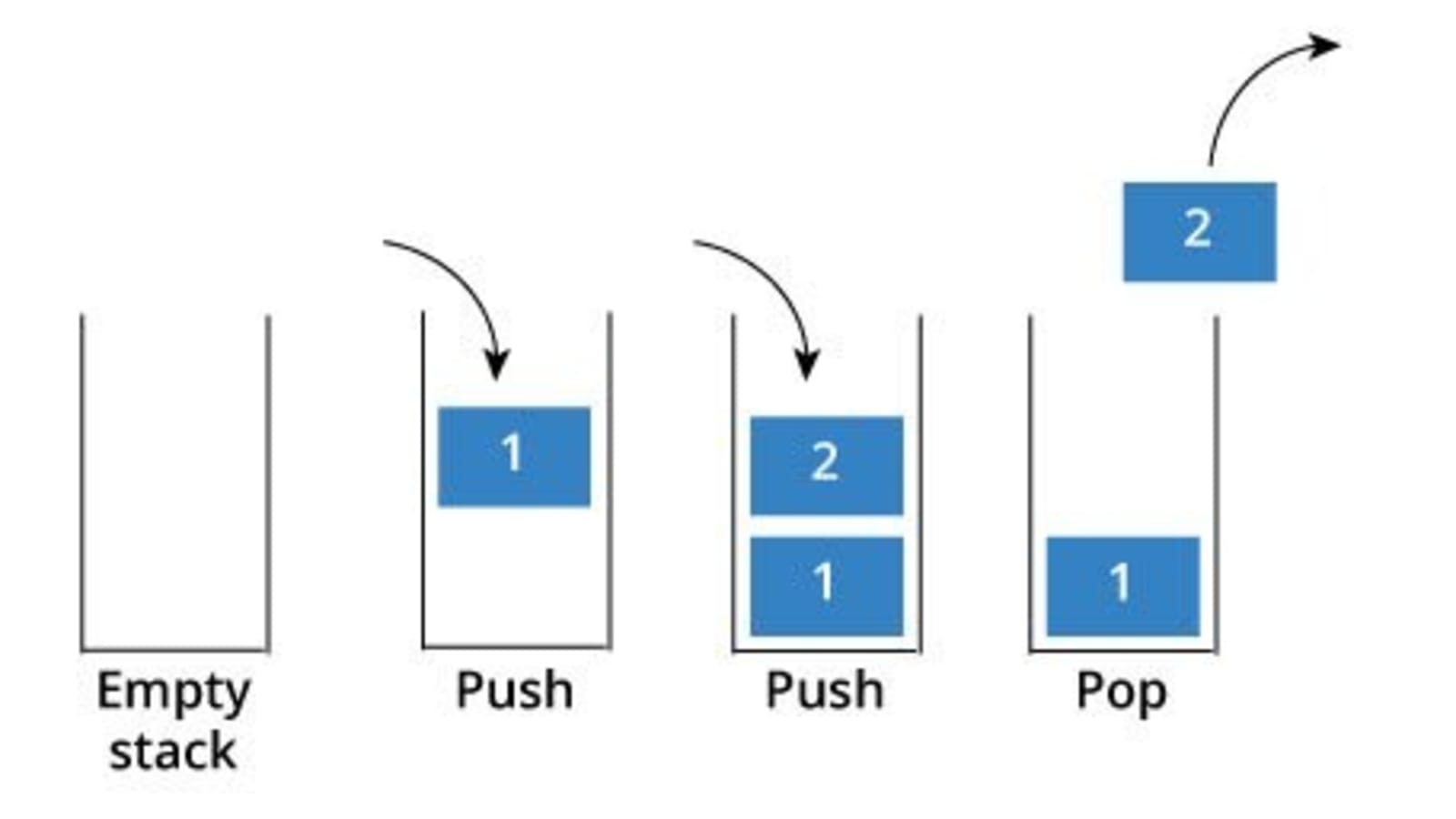
W4\_Qn1

a. The way data is organised and stored in a computer's memory is referred to as its data structure. In other words, it is a generalized array, designed to manage variables of different data types.

b.

The differences between linear data structures and non-linear data structures are:

* Representation: Linear data structures are represented in a linear fashion, while non-linear data structures are represented in a hierarchical or tree-like fashion.
* Traversal: Linear data structures can be traversed linearly, while non-linear data structures require more complex traversal algorithms.
* Non-linear data structures require more complex operations that can be difficult to implement efficiently, whereas linear data structures typically support basic operations.

c. A stack is a linear data structure that follows the LIFO principle, with push and pop operations that add and remove elements from the top of the stack.

d. Linked lists are a type of linear data structure, while stacks are a specialized data structure built on top of a linked list or an array

e. A Stack can be used to evaluate expressions with operands and operators. Stacks can also be used for Backtracking, or checking parenthesis matching in an expression.

W4-Qn2

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 3/24/2023

Date Modified: 3/24/2023

Problem: Week 4 Question 2

Problem Description: Insert a string and print it in reverse.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct reverse

{

char letter;

struct reverse \*next;

};

typedef struct reverse Reverse;

typedef Reverse \*reversePtr;

void pushPop(reversePtr topPtr, char str1[], char str2[], int len);

int main()

{

reversePtr topPtr=NULL;

char str1[20], str2[20]={0};

int len;

printf("Enter a string \n");

gets(str1);

len=strlen(str1);

pushPop(topPtr, str1, str2, len);

printf("The string in reverse order is %s \n", str2);

}

void pushPop(reversePtr topPtr, char str1[], char str2[], int len)

{

reversePtr newPtr, temPtr;

char popValue;

int i=0;

while(len>0)

{

newPtr=(Reverse\*)malloc(sizeof(Reverse));

newPtr->letter=str1[i];

newPtr->next=topPtr;

topPtr=newPtr;

len--;

i++;

}

i=0;

while(topPtr!=NULL)

{

temPtr=topPtr;

popValue=topPtr->letter;

str2[i]=popValue;

topPtr=topPtr->next;

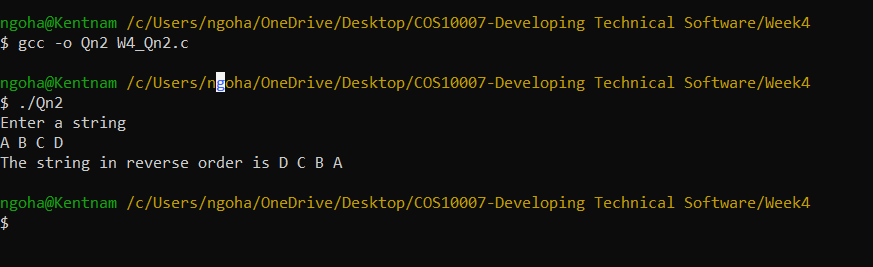
free(temPtr);

i++;

}

}

OUPUT W4\_Qn2



W4\_Qn3

/\*

Unit Code: COS10007

Unit Name: Developing Techinical Software

Student ID: 103488515

Name: Hai Nam Ngo

Date Created: 3/24/2023

Date Modified: 3/24/2023

Problem: Week 4 Question 3

Problem Description: Determine whether a string is a palindrome .

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct reverse

{

char letter;

struct reverse \*next;

};

typedef struct reverse Reverse;

typedef Reverse \*reversePtr;

void pushPop(reversePtr topPtr, char str1[], char str2[], int len);

int main()

{

reversePtr topPtr=NULL;

char str1[20], str2[20]={0};

int len;

printf("Enter a string \n");

gets(str1);

len=strlen(str1);

pushPop(topPtr, str1, str2, len);

printf("The string in reverse order is %s \n", str2);

if(strcmp(str1,str2)==0)

{

printf("The string is a Palindrome.");

}

else

{

printf("The string is not a Palidrome.");

}

}

void pushPop(reversePtr topPtr, char str1[], char str2[], int len)

{

reversePtr newPtr, temPtr;

char popValue;

int i=0;

while(len>0)

{

newPtr=(Reverse\*)malloc(sizeof(Reverse));

newPtr->letter=str1[i];

newPtr->next=topPtr;

topPtr=newPtr;

len--;

i++;

}

i=0;

while(topPtr!=NULL)

{

temPtr=topPtr;

popValue=topPtr->letter;

str2[i]=popValue;

topPtr=topPtr->next;

free(temPtr);

i++;

}

}

OUTPUT for W4\_Qn3

Text

Description automatically generated