Experiment No 1

Aim: Implementation of stack using Array for real-world application

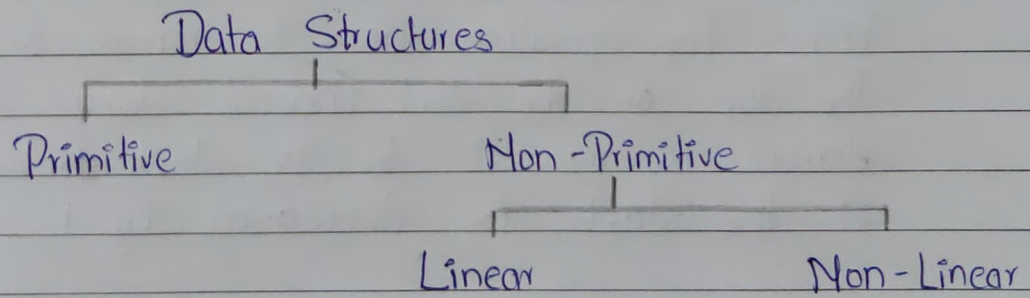
Objective:

1) To introduce the concepts of data structures and analysis procedure.
2) To conceptualize linear data structures and its implementation for various real-world applications.

Theory:

1) Introduction to data structure - Data structure can be defined as the group of elements which provides an efficient way of storing and organising data in the computer so that it can be used efficiently. They are used widely in almost every aspect of computer science.

* Classification of Data structures:



Primitive data structures consists of integer, character and boolean data types. Linear data types consists of array, linked list, stack and queue; whereas, non-linear data types consists of trees and graphs.

2) Introduction to stack -

- A stack is a linear data structure that follows a particular order in which the operations are performed.
- Elements in stack have the same data type and are ordered by when they were added.
- In stack, top element is the only accessible element.
- The order of stack is also known as Last In First Out (LIFO)

3) Various operations (PUSH, POP, PEEP, CHANGE, DISPLAY, etc)

PUSH - Push operation refers to inserting an element in the stack. Since there's only one position at which the new element can be inserted - TOP of stack, the new element is inserted at the top of stack

POP - Pop operation refers to remove an element from the top of the stack (newest element in stack). The element is removed to the stack container and the size of the stack is decreased by 1.

PEEP - Peep operation is stack function that returns the value of the topmost element of the stack without deleting that element from the stack.

CHANGE - Change operation in stack function that user can change or update the contents of the specific element.

DISPLAY - The display() function displays all the elements in the stack. It uses a for loop to do so. If there is no elements in the stack, then stack is empty.

4) Algorithm :

PUSH - This inserts an element x to the top of the stack which is represented by a vector s containing N elements with a pointer TOP denoting the top element in the stack.

1. [Check for stack overflow]
   If TOP >= N
   then Write ('Stack overflow')
   Return
2. [Increment TOP]
   TOP ← TOP + 1
3. [insert element]
   s[TOP] ← x
4. Finished

POP - This removes the top element from a stack which is represented by vector s and returns this element. TOP is a pointer to the top element of the stack.

1. [Check for underflow on stack]
   If TOP = 0
   then Write ('Stack underflow on POP')
   take action in response to underflow
   Exit
2. [Decrement pointer]
   TOP ← TOP - 1
3. [Return former top element of stack]
   Return (S[TOP + 1])

PEEP - Given a vector consisting of N element representing a sequentially allocated stack. and a pointer TOP denoting the top element of the stack. this function returns the value of the top most element from the stack.

1. [Check for stack underflow]
   If TOP - i + 1 <= 0
   then write ('Stack underflow on Peep')
   take action in response to underflow
   Exit
2. Return ith element from top of the stack
   Return (S[TOP - I + i])

- CHANGE - This changes the value of i^{th} element from the top of the stack to the value contained in x.

1. [Check for stack underflow]
   If TOP - i + 1 <= 0
   then write ('stack underflow on change')
   Return
2. Change i^{th} element from top of the stack.
   S[TOP - i + 1] ← x
3. [Finished]
   Return

- DISPLAY -

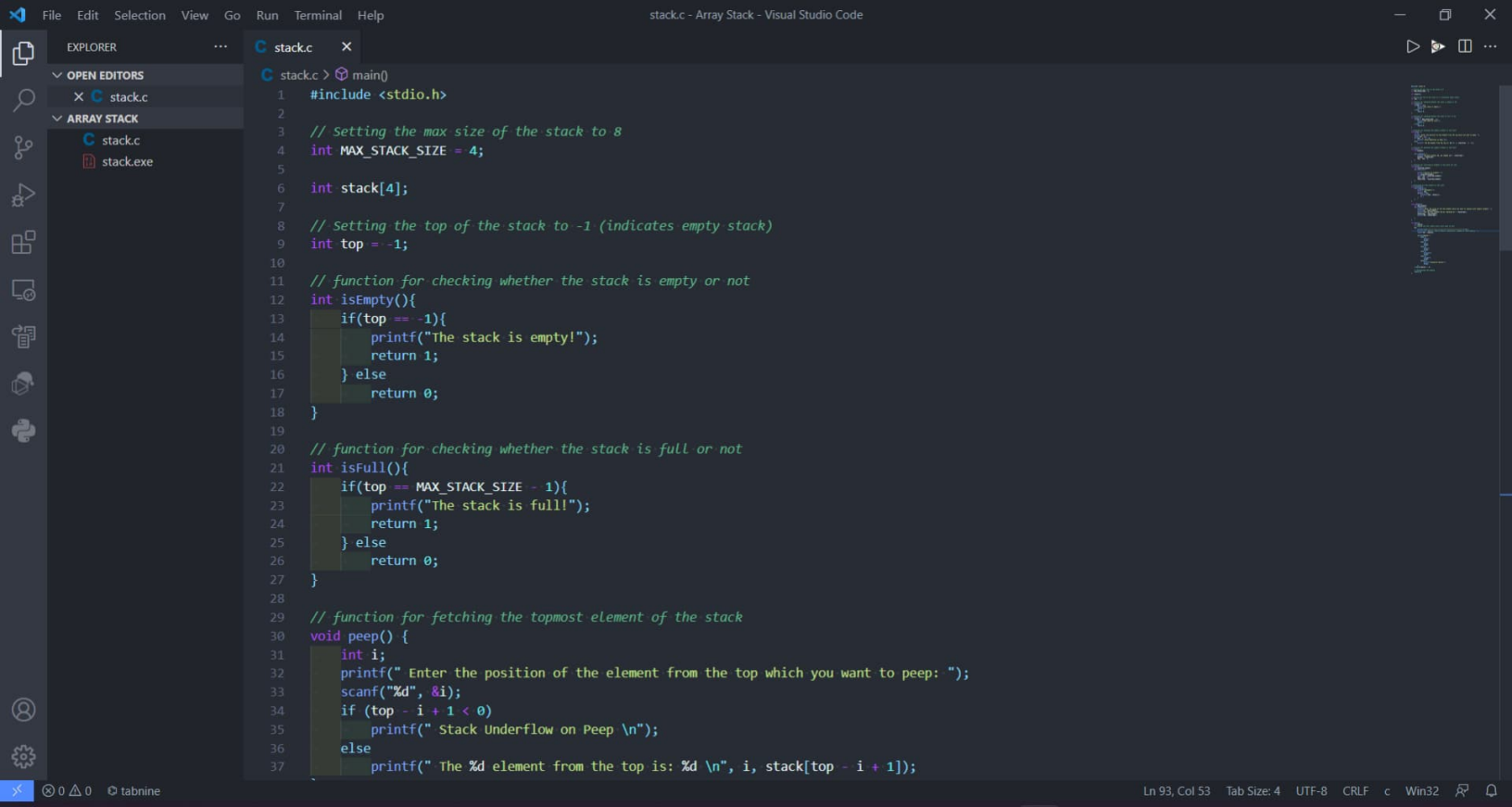
1. Check whether stack is empty (TOP == -1)
2. If it is empty, then display "Stack is empty!!" and terminate the function.
3. If it is not empty, then define a variable 'i' and initialize with top. Display stack[i] value and decrement i value by one (i--).
4. Repeat above step until i value becomes '0'

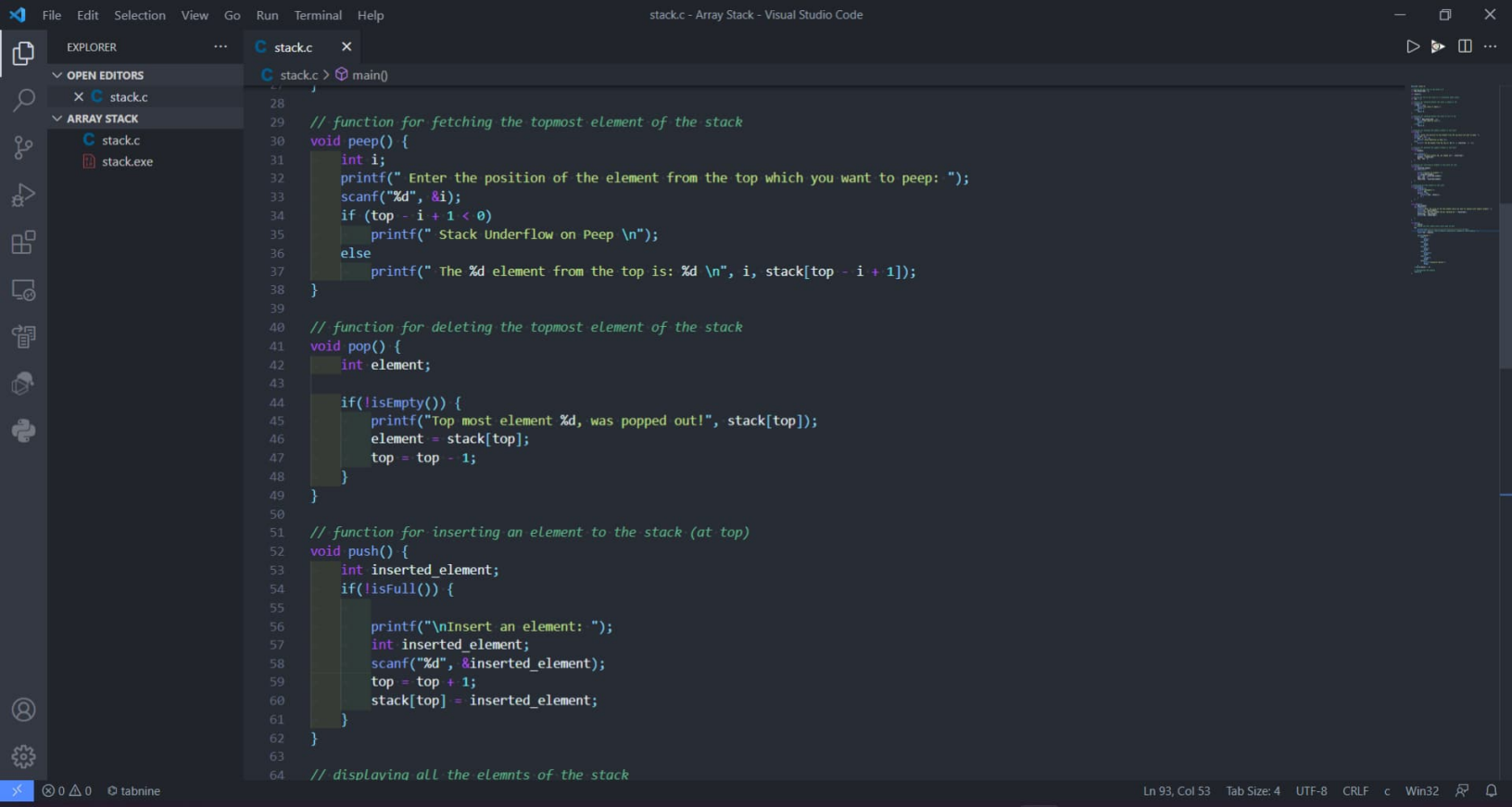
5) Examples:

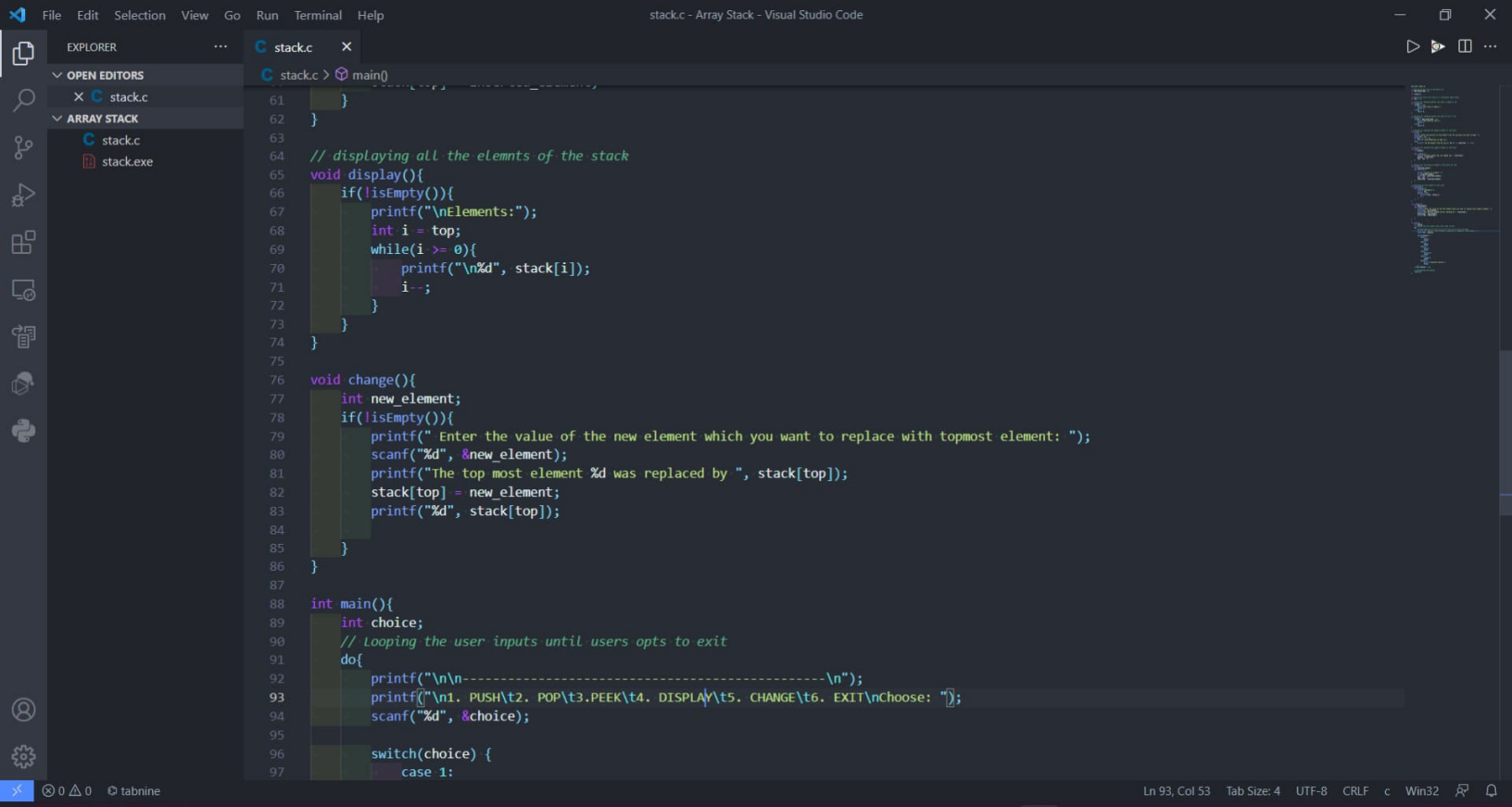
① Undo/Redo operation in applications, file management systems, etc.
② Stack of books in a cupboard.

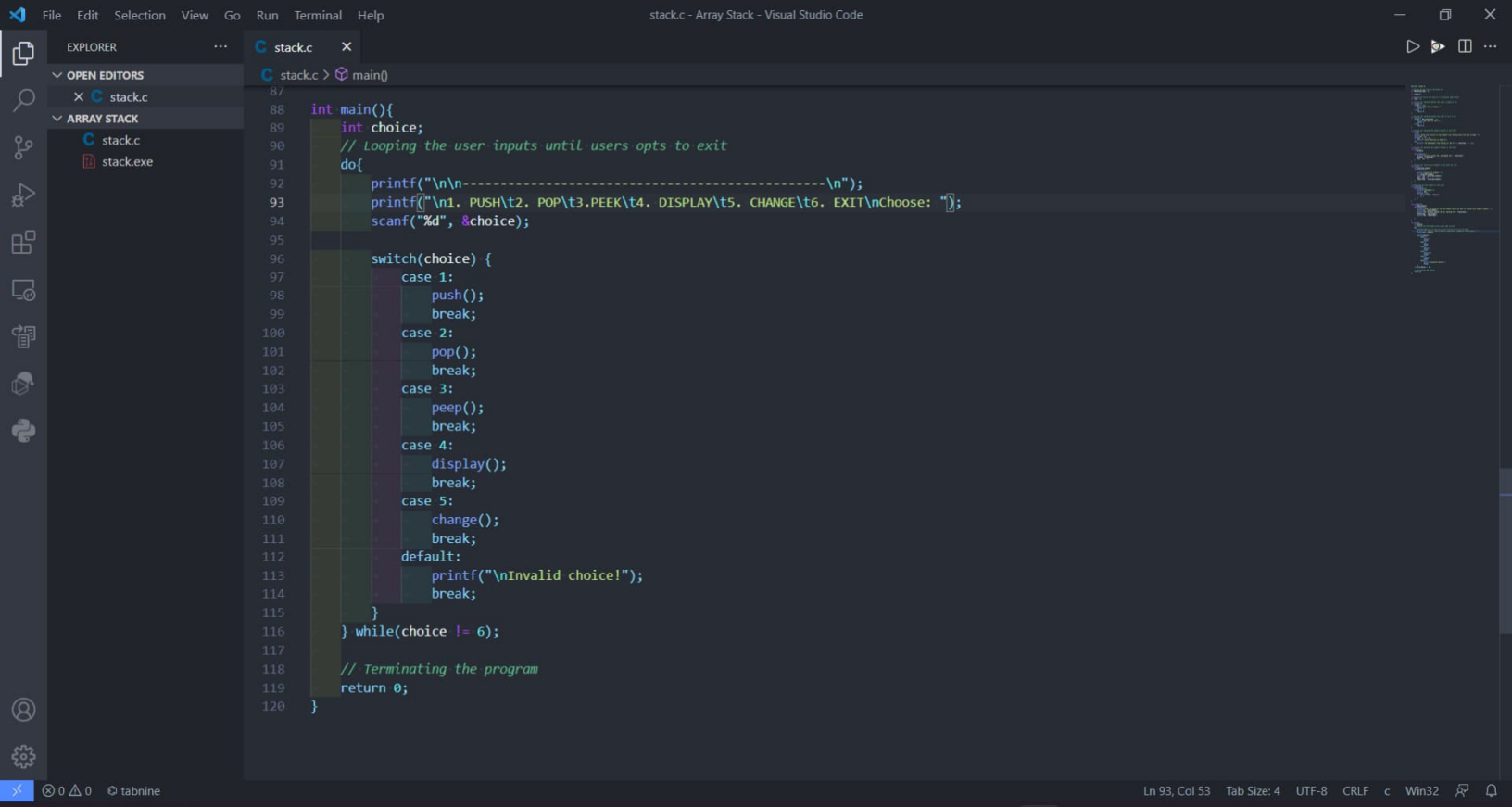
Conclusion: In this experiment, we learnt about stacks in data structures. We implemented stacks and performed its basic operations like Push, Pop, Peep, change and display. ~~operations in the stack~~ We came across the examples where stack is implemented in real world.

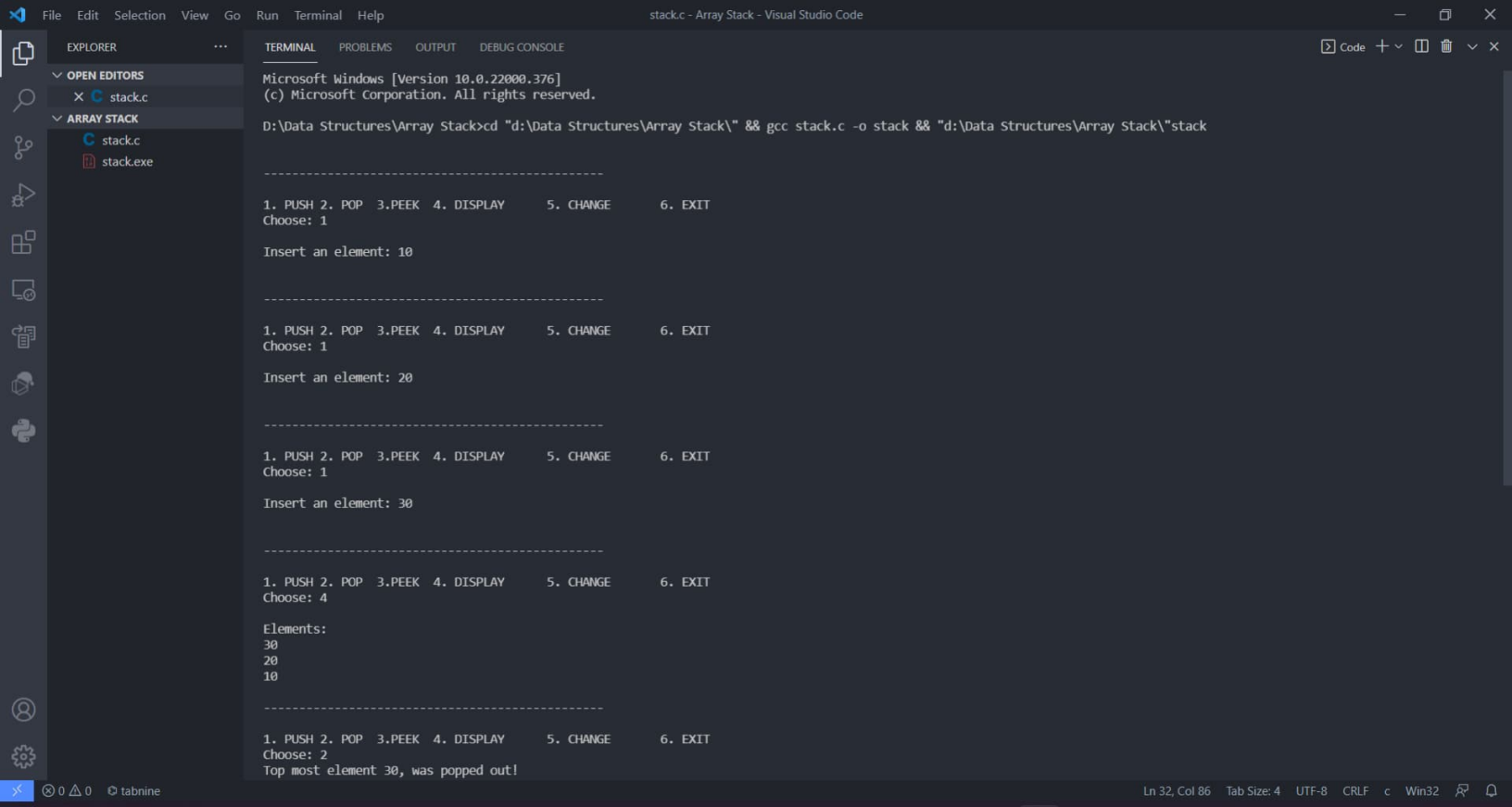
Output: Apply the concepts of stack for real-world application.


File Edit Selection View Go Run Terminal Help
stack.c - Array Stack - Visual Studio Code
EXPLORER
OPEN EDITORS
stack.c
ARRAY STACK
stack.c
stack.exe
stack.c
stack.c > main()
#include <stdio.h>
// Setting the max size of the stack to 8
int MAX_STACK_SIZE = 4;
int stack[4];
// Setting the top of the stack to -1 (indicates empty stack)
int top = -1;
// function for checking whether the stack is empty or not
int isEmpty(){
if(top == -1){
printf("The stack is empty!");
return 1;
} else
return 0;
}
// function for checking whether the stack is full or not
int isFull(){
if(top == MAX_STACK_SIZE - 1){
printf("The stack is full!");
return 1;
} else
return 0;
}
// function for fetching the topmost element of the stack
void peep() {
int i;
printf(" Enter the position of the element from the top which you want to peep: ");
scanf("%d", &i);
if (top - i + 1 < 0)
printf(" Stack Underflow on Peep \n");
else
printf(" The %d element from the top is: %d \n", i, stack[top - i + 1]);
0 0 tabnine
Ln 93, Col 53 Tab Size: 4 UTF-8 CRLF c Win32


File Edit Selection View Go Run Terminal Help
stack.c - Array Stack - Visual Studio Code
EXPLORER
OPEN EDITORS
stack.c
ARRAY STACK
stack.c
stack.exe
stack.c
stack.c > main()
// function for fetching the topmost element of the stack
void peep() {
int i;
printf(" Enter the position of the element from the top which you want to peep: ");
scanf("%d", &i);
if (top - i + 1 < 0)
printf(" Stack Underflow on Peep \n");
else
printf(" The %d element from the top is: %d \n", i, stack[top - i + 1]);
}
// function for deleting the topmost element of the stack
void pop() {
int element;
if(!isEmpty()) {
printf("Top most element %d, was popped out!", stack[top]);
element = stack[top];
top = top - 1;
}
}
// function for inserting an element to the stack (at top)
void push() {
int inserted_element;
if(!isFull()) {
printf("\nInsert an element: ");
int inserted_element;
scanf("%d", &inserted_element);
top = top + 1;
stack[top] = inserted_element;
}
}
// displaying all the elemnts of the stack
0 0
tabnine
Ln 93, Col 53
Tab Size: 4
UTF-8
CRLF
c
Win32


File Edit Selection View Go Run Terminal Help
stack.c - Array Stack - Visual Studio Code
EXPLORER
OPEN EDITORS
stack.c
ARRAY STACK
stack.c
stack.exe
stack.c
stack.c > main()
}
}
// displaying all the elemnts of the stack
void display(){
if(!isEmpty()){
printf("\nElements:");
int i = top;
while(i >= 0){
printf("\n%d", stack[i]);
i--;
}
}
}
void change(){
int new_element;
if(!isEmpty()){
printf(" Enter the value of the new element which you want to replace with topmost element: ");
scanf("%d", &new_element);
printf("The top most element %d was replaced by ", stack[top]);
stack[top] = new_element;
printf("%d", stack[top]);
}
}
int main(){
int choice;
// Looping the user inputs until users opts to exit
do{
printf("\n\n---\n");
printf("\n1. PUSH\t2. POP\t3.PEEK\t4. DISPLAY\t5. CHANGE\t6. EXIT\nChoose: ");
scanf("%d", &choice);
switch(choice) {
case 1:
tabnine
Ln 93, Col 53
Tab Size: 4
UTF-8
CRLF
c
Win32


File Edit Selection View Go Run Terminal Help
stack.c - Array Stack - Visual Studio Code
EXPLORER
OPEN EDITORS
stack.c
ARRAY STACK
stack.c
stack.exe
stack.c
stack.c > main()
int main(){
int choice;
// Looping the user inputs until users opts to exit
do{
printf("\n\n---\n");
printf("\n1. PUSH\t2. POP\t3.PEEK\t4. DISPLAY\t5. CHANGE\t6. EXIT\nChoose: ");
scanf("%d", &choice);
switch(choice) {
case 1:
push();
break;
case 2:
pop();
break;
case 3:
peep();
break;
case 4:
display();
break;
case 5:
change();
break;
default:
printf("\nInvalid choice!");
break;
}
} while(choice != 6);
// Terminating the program
return 0;
}
0 0
tabnine
Ln 93, Col 53
Tab Size: 4
UTF-8
CRLF
c
Win32


File Edit Selection View Go Run Terminal Help
stack.c - Array Stack - Visual Studio Code
EXPLORER
OPEN EDITORS
stack.c
ARRAY STACK
stack.c
stack.exe
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
Code
Microsoft Windows [Version 10.0.22000.376]
(c) Microsoft Corporation. All rights reserved.
D:\Data Structures\Array Stack>cd "d:\Data Structures\Array Stack\" && gcc stack.c -o stack && "d:\Data Structures\Array Stack\"stack
1. PUSH 2. POP 3.PEEK 4. DISPLAY 5. CHANGE 6. EXIT
Choose: 1
Insert an element: 10
1. PUSH 2. POP 3.PEEK 4. DISPLAY 5. CHANGE 6. EXIT
Choose: 1
Insert an element: 20
1. PUSH 2. POP 3.PEEK 4. DISPLAY 5. CHANGE 6. EXIT
Choose: 1
Insert an element: 30
1. PUSH 2. POP 3.PEEK 4. DISPLAY 5. CHANGE 6. EXIT
Choose: 4
Elements:
30
20
10
1. PUSH 2. POP 3.PEEK 4. DISPLAY 5. CHANGE 6. EXIT
Choose: 2
Top most element 30, was popped out!
tabnine
Ln 32, Col 86 Tab Size: 4 UTF-8 CRLF c Win32

```
Insert an element: 30


------------------------------------------------

1. PUSH 2. POP  3.PEEK  4. DISPLAY      5. CHANGE       6. EXIT
Choose: 4

Elements:
30
20
10

------------------------------------------------

1. PUSH 2. POP  3.PEEK  4. DISPLAY      5. CHANGE       6. EXIT
Choose: 2
Top most element 30, was popped out!

------------------------------------------------

1. PUSH 2. POP  3.PEEK  4. DISPLAY      5. CHANGE       6. EXIT
Choose: 4

Elements:
20
10

------------------------------------------------

1. PUSH 2. POP  3.PEEK  4. DISPLAY      5. CHANGE       6. EXIT
Choose: 3
 Enter the position of the element from the top which you want to peep: 1
 The 1 element from the top is: 20


------------------------------------------------

1. PUSH 2. POP  3.PEEK  4. DISPLAY      5. CHANGE       6. EXIT
Choose: 5
 Enter the value of the new element which you want to replace with topmost element: 44
The top most element 20 was replaced by 44

------------------------------------------------
```