|  |  |
| --- | --- |
|  | // C program to implement multiple stack operation using SLL  #include <stdio.h>  #include <stdlib.h> |
|  | #include <limits.h> |
|  |  |
|  | #define CAPACITY 10000 |
|  |  |
|  | struct stack |
|  | { |
|  | int data; |
|  | struct stack \*next; |
|  | } \*top; |
|  |  |
|  | int size = 0; |
|  | void push(int element); |
|  | int pop(); |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | int choice, data; |
|  |  |
|  | while(1) |
|  | { |
|  | printf("------------------------------------\n"); |
|  | printf(" STACK IMPLEMENTATION PROGRAM \n"); |
|  | printf("------------------------------------\n"); |
|  | printf("1. Push\n"); |
|  | printf("2. Pop\n"); |
|  | printf("3. Size\n"); |
|  | printf("4. Exit\n"); |
|  | printf("------------------------------------\n"); |
|  | printf("Enter your choice: "); |
|  |  |
|  | scanf("%d", &choice); |
|  |  |
|  | switch(choice) |
|  | { |
|  | case 1: |
|  | printf("Enter data to push into stack: "); |
|  | scanf("%d", &data); |
|  | push(data); |
|  | break; |
|  |  |
|  | case 2: |
|  | data = pop() |
|  | if (data != INT\_MIN) |
|  | printf("Data => %d\n", data); |
|  | break; |
|  |  |
|  | case 3: |
|  | printf("Stack size: %d\n", size); |
|  | break; |
|  |  |
|  | case 4: |
|  | printf("Exiting from app.\n"); |
|  | exit(0); |
|  | break; |
|  |  |
|  | default: |
|  | printf("Invalid choice, please try again.\n"); |
|  | } |
|  |  |
|  | printf("\n\n"); |
|  | } |
|  |  |
|  |  |
|  | return 0 |
|  | } |
|  |  |
|  | void push(int element) |
|  | { |
|  |  |
|  | if (size >= CAPACITY) |
|  | { |
|  | printf("Stack Overflow, can't add more element to stack.\n"); |
|  | return; |
|  | } |
|  |  |
|  |  |
|  | struct stack \* newNode = (struct stack \*) malloc(sizeof(struct stack)); |
|  |  |
|  |  |
|  | newNode->data = element; |
|  |  |
|  |  |
|  | newNode->next = top; |
|  |  |
|  |  |
|  | top = newNode; |
|  |  |
|  |  |
|  | size++; |
|  |  |
|  | printf("Data pushed to stack.\n"); |
|  | } |
|  |  |
|  |  |
|  |  |
|  | int pop() |
|  | { |
|  | int data = 0; |
|  | struct stack \* topNode; |
|  |  |
|  |  |
|  | if (size <= 0 || !top) |
|  | { |
|  | printf("Stack is empty.\n"); |
|  |  |
|  |  |
|  | return INT\_MIN; |
|  | } |
|  |  |
|  | topNode = top; |
|  |  |
|  |  |
|  | data = top->data; |
|  |  |
|  | top = top->next; |
|  |  |
|  |  |
|  | free(topNode); |
|  |  |
|  |  |
|  | size--; |
|  |  |
|  | return data; |
|  | } |