|  |  |
| --- | --- |
| Program No: 01 | Date: 23/07/2025 |
| Program Title : Write programs to demonstrate the use of storage classes (local variable, global variable, static variable, register variable) in C. | |
| /\*Program 1 USE DIFFERENT STORAGE CLASSES (LOCAL,GLOBAL,STATIC,REGISTER)  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 23/07/2025  \*/  #include<stdio.h>  int a=10,b=10; //global var  void display() {  int c=10;  static int d=10; //static var  printf("local under increment: %d\t static under increment:%d\n",c++,d++);  }  int main(){  register int e=11; //register var  int b=5; //local var  printf("Global variable: %d\nLocal variable: %d\n",a,b);  printf("Register variable: %d\n",e);  printf("Static variable: \n");  display();  display();  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 02 | Date: 23/07/2025 |
| Program Title : Use a menu-driven program to insert, search, delete and sort elements in an array using functions (use global variables). | |
| /\*PROGRAM-2 A MENU FOR ARRAY OPERATIONS(INSERT,DELETE,DISPLAY,SEARCH,SORT) USING GLOBAL VARIABLE  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 23/07/2025  \*/  #include<stdio.h>  int stack[5]; //declaring stack  int top = -1; //declaring variable positionof top element  int insert(int e) { //Function to insert element into stack  if (top + 1 == 5) {  printf("Error: Stack is Full");  }  else {  stack[++top] = e;  }  return top;  }  int erase() { //function to delete top element  if (top == -1){  printf("Error: Stack is empty");  }  else {  printf("\n %d",stack[top--]);  }  return top;  }  void search(int b, int a[5], int top){ //function to search elements  int isfound = 0, i;  for (i = 0; i <= top; i++) {  if (b == a[i]) {  isfound = 1;  printf("Element found at [%d] position. \n", i);  }  }  if (isfound == 0)  printf("element not found");  }    void display() { //function to display the elements in stack  if (top == -1)  printf("Empty Stack");  else {  int i;  for (i = 0; i <= top; i++){  printf("%d \t",stack[i]);  }  printf("\n");  }  }  void sort(){ //function to sort the stack  int i, j, temp;  for (i = 0; i < 5; i++) {  for (j = 0; j < 5; j++) {  if(stack[i] < stack[j]) {  temp = stack[i];  stack[i] = stack[j];  stack[j] = temp;  }  }  }  }  int menu() { //function for menu  int ch;  printf("\n INSERT-1 \n DELETE-2 \n DISPLAY-3 \n SEARCH-4 \n SORT-5 \n EXIT -6 \n Enter your choice: ");  scanf("%d", &ch);  return ch;  }  void processStack() { //working of menu  int ch, b;  for (ch = menu(); ch != 6; ch = menu()) {  switch(ch){  case 1: //insert  printf("Enter the value to insert: ");  scanf("%d", &ch);  insert(ch);  break;  case 2: //delete  erase();  break;  case 3: //display  display();  break;  case 4: //search  printf("Enter the value to search: ");  scanf("%d",&b);  search(b, stack, top);  break;  case 5: // sort  sort();  break;  default: //any other options  printf("Error: Wrong Choice");  break;  }  }  }  int main() {  processStack();  return 0;  } | |
| Output | |
| Program No: 03 | Date: 23/07/2025 |
| Program Title : Use a menu-driven program to insert, search, delete and sort elements in an array using functions (use only local variables). | |
| /\*PROGRAM-3 A MENU FOR ARRAY OPERATIONS(INSERT,DELETE,DISPLAY,SEARCH,SORT) USING LOCAL VARIABLE  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 23/07/2025  \*/  #include <stdio.h>  int insert(int a[5], int pos, int e) { //function to insert an element  if (pos + 1 == 5) {  printf("Error: Array is Full\n");  }  else {  a[++pos] = e;  }  return pos;  }  int erase(int a[5], int pos) { //function to delete top element  if (pos == -1) {  printf("Error: Array is Empty\n");  }  else {  printf("Deleted element: %d\n", a[pos--]);  }  return pos;  }  void display(int a[5], int pos){ //function to display entire array  if (pos == -1) {  printf("Error: Array is Empty\n");  }  else {  int i;  for (i = 0; i <= pos; i++) {  printf("%d\t", a[i]);  }  printf("\n");  }  }  void search(int b, int a[5], int pos){ //function to search for an element and display its index  int isfound = 0, i;  for (i = 0; i <= pos; i++) {  if(b==a[i]) {  isfound=1;  printf("Element found at [%d] position. \n",i);  }  }  if (isfound == 0)  printf("element not found");  }  void sort(int a[5]){ //function to sort the stack  int i, j, temp;  for (i = 0; i < 5; i++) {  for (j = 0; j < 5; j++) {  if(a[i] < a[j]) {  temp = a[i];  a[i] = a[j];  a[j] = temp;  }  }  }  }  int menu() { //function to create menu interface  int ch;  printf("\nInsert - 1\nDelete - 2\nDisplay - 3\nSearch - 4\nSort - 5\nExit - 6\nEnter your choice: ");  scanf("%d", &ch);  return ch;  }  void processArray() { //working of menu  int a[5], pos = -1, b;  int ch, value;  for (ch = menu(); ch != 6; ch = menu()) {  switch (ch) {  case 1:  printf("Enter value to insert: ");  scanf("%d", &value);  pos = insert(a, pos, value);  break;  case 2:  pos = erase(a, pos);  break;  case 3:  display(a, pos);  break;  case 4:  printf("Enter the element to search: ");  scanf("%d",&b);  search(b,a,pos);  break;  case 5:  sort(a);  break;  default:  printf("Error: Wrong Choice.\n");  }  }  }  int main() {  processArray();  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 04 | Date: 23/07/2025 |
| Program Title : Search for all the occurrences of an element in an integer array (positions). | |
| /\*PROGRAM-4 ARRAY SEARCH  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 23/07/2025  \*/  #include<stdio.h>  void search(int b,int a[5]){ //function for search function  int isfound = 0, i;  for (i = 0; i < 5; i++){  if (b == a[i]) {  isfound = 1;  printf("Element found at [%d] position. \n",i); //displays index  }  }  if (isfound = 0)  printf("element not found");  }  int main() {  int a[5], ch, i;  for (i = 0; i < 5; i++) {  printf("Enter the [%d] element : ",i);  scanf("%d",&a[i]);  }  for (i = 0; i < 5; i++) {  printf("%d \t",a[i]);  }  printf("\n");  printf("Enter the element to search: ");  scanf("%d", &ch);  search(ch,a);  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 05 | Date: 23/07/2025 |
| Program Title : Sort the array elements in ascending order (minimum three functions: read, disp and sort). | |
| /\*PROGRAM-5 SORT ARRAY IN ASC WITH ALTEAST 3 FUNCTIONS - READ DISP SORT  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 23/07/2025  \*/  #include<stdio.h>  void read(int n,int arr[n]){ //function to insert elements in array  int i;  for (i = 0; i < n; i++) {  printf("Enter the value for %d :",i);  scanf("%d", &arr[i]);  }  }  void disp(int n,int arr[n]) { //function to display all elemenyts in array  int i;  for (i = 0; i < n; i++) {  printf("%d\t", arr[i]);  }  }  void sort(int n,int arr[n]) { //function for sorting the elements in array in ascending order  int i, j, temp;  for (i = 0; i < n; i++) {  for (j = 0; j < n; j++) {  if (arr[i] < arr[j]) {  temp = arr[i];  arr[i] = arr[j];  arr[j] = temp;  }  }  }  }  int menu() { //funtion for menu interface  int ch;  printf("\nREAD-1\nSORT-2\nDISPLAY-3\nEXIT-4\nENTER YOUR CHOICE: ");  scanf("%d", &ch);  return ch;  }  void processArray(int n,int arr[n]) { //working of menu  int ch;  for (ch = menu(); ch != 4; ch = menu()) {  switch(ch) {  case 1:  read(n,arr);  break;  case 2:  sort(n,arr);  break;  case 3:  disp(n,arr);  break;  default:  printf("Errror: Wrong Choice\n");  break;  }  }  }  int main() {  int n;  printf("Enter limit of Array:");  scanf("%d",&n);  int arr[n];  processArray(n,arr);  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 06 | Date: 23/07/2025 |
| Program Title : Display the array elements in the same order using a recursive function. | |
| /\*PROGRAM-6 DISPLAY ARRAY USING RECURSIVE FUNCTION  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 25/07/2025  \*/  #include<stdio.h>  int j = 0, arr[10];  void display() { //display function as a recursive function  if (j < 10) {  printf("%d\t",arr[j]);  j++;  display();  }  If (j == 10)  j = 0;  }  int main() {  int i=0;  for (i = 0; i < 10; i++) {  printf("Enter the value for %d:", i);  scanf("%d", &arr[i]);  }  display();  return 0;  } | |
| Output | |
| Program No: 07 | Date: 23/07/2025 |
| Program Title : Display array elements in reverse order using a recursive function. | |
| /\*PROGRAM-7 DISPLAY ARRAY INN REVERSE USING RECURSIVE FUNCTION  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 26/07/2025  \*/  #include<stdio.h>  int j = 10, arr[10];  void display() { //function to display elements in reverse using recursive function  if (j > 0) {  j--;  printf("%d\t",arr[j]);  display();  if (j == 0)  j = 10;  }  }  int main() {  int i = 0;  for (i = 0; i < 10; i++) {  printf("Enter the value for %d:", i);  scanf("%d", &arr[i]);  }  display();  return 0;  } | |
| Output | |
| Program No: 08 | Date: 23/07/2025 |
| Program Title : Write a program to Perform the addition of two matrix and Subtraction of one matrix from another. | |
| /\*PROGRAM-8 MATRIX ADDITION AND SUBTRACTION  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 25/07/2025  \*/  #include<stdio.h>  int a[10][10], b[10][10], m, n;  void insert(int e[10][10],int m, int n){ //function to insert values into the matrix  int i, j;  for (i = 0; i < m; i++){  for (j = 0; j < n; j++){  printf("Enter the value of [%d] [%d] \: ", i, j);  scanf("%d", &e[i][j]);  }  }  }  void print(int p[10][10],int m, int n){ //function to print a matrix  int i, j;  for (i = 0; i < m; i++) {  for (j = 0; j < n; j++) {  printf("%d\t",p[i][j]);  }  printf("\n");  }  }  void add() { //funtion to add two matrices  int sum[10][10], i, j;  for (i = 0; i < m; i++) {  for (j = 0; j < n; j++) {  sum[i][j]=a[i][j]+b[i][j];  }  }  print(sum,m,n);  }  void diffn() { //function to subtract a matrix from another  int dif[10][10], i, j;  for (i = 0; i < m; i++) {  for (j = 0; j < n; j++) {  dif[i][j]=a[i][j]-b[i][j];  }  }  print(dif,m,n);  }  int main() {  printf("Enter the dimensions of the Matrix: ");  scanf("%d%d",&m,&n);  printf("Enter the First matrix :\n");  insert(a,m,n);  printf("Enter the Second matrix : \n");  insert(b,m,n);  printf("The first matrix is : \n");  print(a,m,n);  printf("The Second matrix is : \n");  print(b,m,n);  printf("The sum of matrices is : \n");  add(m,n);  printf("The difference of matrices is : \n");  diffn(m,n);  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 09 | Date: 23/07/2025 |
| Program Title : Write a program to perform multiplication of two matrix. | |
| /\*PROGRAM-9 MATRIX MULTIPLICATION  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 25/07/2025  \*/  #include<stdio.h>  int a[10][10],b[10][10], m, n, p;  void insert(int e[10][10], int m, int n) { //function to insert values in the matrix  int i, j;  for (i = 0; i < m; i++) {  for(j=0;j<n;j++){  printf("Enter the value of [%d] [%d]: ", i, j);  scanf("%d",&e[i][j]);  }  }  }  void print(int p[10][10], int m, int n) { //funtion to display the matrix  int i, j;  for (i = 0; i < m; i++) {  for (j = 0; j < n; j++) {  printf("%d\t",p[i][j]);  }  printf("\n");  }  }  void multi() { //function to multipy two matrices  int prod[10][10], i, j, k;  for ( i = 0; i < m; i++) { //declaring intial value of elements of product to be zero  for (j = 0; j < p; j++) {  prod[i][j]=0;  }  }  for (i = 0; i < m; i++) {  for (j = 0; j < p; j++) {  for (k = 0; k < n; k++) {  prod[i][j] += a[i][k] \* b[k][j];  }  }  }  print(prod,m,p);  }  int main() {  printf("Enter the dimensions of first Matrix: ");  scanf("%d%d",&m,&n);  printf("Enter the First matrix :\n");  insert(a,m,n);  printf("Enter the number of columns for the Second Matrix: ");  scanf("%d", &p);  printf("Enter the Second matrix : \n");  insert(b,n,p);  printf("The first matrix is : \n");  print(a,m,n);  printf("The Second matrix is : \n");  print(b,n,p);  printf("The product of matrices is : \n");  multi();  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 10 | Date: 23/07/2025 |
| Program Title : Write a program to find the transpose of a matrix. | |
| /\*PROGRAM-10 MATRIX TRANSPOSE  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 25/07/2025  \*/  #include<stdio.h>  int a[10][10], t[10][10];  void print(int e[10][10], int m, int n) { //function to print the matrix.  int i, j;  for (i = 0; i < m; i++) {  for (j = 0; j < n; j++) {  printf("%d\t", e[i][j]);  }  printf("\n");  }  }  void transpose(int m, int n) { //function to find transpose of matrix  int i,j;  for (i = 0; i < m; i++) {  for(j = 0; j < n; j++) {  t[j][i] = a[i][j];  }  }  printf("The Transpose of the matrix is : \n");  print(t,n,m);  }  int main() {  int m,n;  printf("Enter the dimensions of the Matrix:");  scanf("%d%d", &m, &n);  int i, j;  for (i = 0; i < m; i++) {  for (j = 0; j < n; j++) {  printf("Enter the value of [%d] [%d]: ", i, j);  scanf("%d",&a[i][j]);  }  }  printf("The matrix is : \n");  print(a,m,n);  transpose(m,n);  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 11 | Date: 23/07/2025 |
| Program Title : Write a program to find the Determinant of a matrix (2x2 and 3x3). | |
| /\* PROGRAM-11 DETERMINANT OF A MATRIX  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 26/07/2025  \*/  #include <stdio.h>  int main() {  int size, a[3][3], i, j;  float det;  printf("Enter the size of the square matrix (2 or 3): ");  scanf("%d",&size);  if (size != 2 && size != 3) {  printf("Only 2x2 and 3x3 matrices are supported.\n");  return 1;  }  printf("Enter the elements of the matrix:\n");  for (i = 0; i < size; i++) {  for (j = 0; j < size; j++) {  printf("a[%d][%d]: ", i, j);  scanf("%d", &a[i][j]);  }  }  printf("The matrix is:\n");  for (i = 0;I < size; i++) {  for (j = 0; j < size; j++) {  printf("%d\t", a[i][j]);  }  printf("\n");  }  if (size == 2) {  det=a[0][0] \* a[1][1] - a[0][1] \* a[1][0]; // For 2x2: |A| = ad - bc  } else if (size==3) {  det = a[0][0] \* (a[1][1] \* a[2][2] - a[1][2] \* a[2][1]) //expansion of formula - a[0][1] \* (a[1][0] \* a[2][2] - a[1][2] \* a[2][0])  + a[0][2] \* (a[1][0] \* a[2][1] - a[1][1] \* a[2][0]);  }  printf("Determinant of the matrix = %.2f\n", det);  return 0;  } | |
| Output | |
| Program No: 12 | Date: 23/07/2025 |
| Program Title : Implement stack operations using arrays. | |
| /\*PROGRAM-12 STACK OPERATIONS USING ARRAY  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 26/07/2025  \*/  #include <stdio.h>  int push(int stack[5], int top, int e) { //function to push elements onto stack  if (top + 1 == 5) {  printf("Error: Stack is Full\n");  } else {  stack[++top] = e;  printf("Pushed\n");  }  return top;  }  int pop(int stack[5], int top){ //function to pop the top element from stack  if (top == -1){  printf("Error: Stack is Empty\n");  } else {  printf("Popped element: %d\n", stack[top--]);  }  return top;  }  void peek(int stack[5], int top) { //function to peek the top element in stack  if (top == -1) {  printf("Stack is Empty\n");  }else{  printf("Top element: %d \n",stack[top]);  }  }  int menu() { //function for menu interface  int ch;  printf("\nPush - 1\nPop - 2\nPeek - 3\nExit - 4\nEnter your choice: ");  scanf("%d", &ch);  return ch;  }  void processStack() { //working of menu  int stack[5], top = -1;  int ch, value;  for (ch = menu(); ch != 4; ch = menu()) {  switch (ch) {  case 1:  printf("Enter value to insert: ");  scanf("%d", &value);  top = push(stack, top, value);  break;  case 2:  top = pop(stack, top);  break;  case 3:  peek(stack, top);  break;  default:  printf("Error: Wrong Choice.\n");  }  }  }  int main() {  processStack();  return 0;  } | |
| Output | |
| Program No: 13 | Date: 23/07/2025 |
| Program Title : Read a String and Just print it in the reverse order. | |
| /\* PROGRAM-13 STRING REVERSAL  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 27/07/2025  \*/  #include<stdio.h>  void str\_rev(char a[20]){ //program to reverse a string  int i=0;  while (a[i] != '\0') {  i++;  }  for ( ; i >= 0; i--) {  printf("%c", a[i]);  }  }  int main() {  char a[20];  printf("Enter a String: ");  gets(a);  str\_rev(a);  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 14 | Date: 23/07/2025 |
| Program Title : Read a String and Reverse the string in the same array itself. | |
| /\* PROGRAM-14 STRING REVERSAL IN THE SAME ARRRAY  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 27/07/2025  \*/  #include <stdio.h>  #include <string.h>  void reverse(char str[20]) { //function to reverse the string  int i = 0, j;  char temp;  j = strlen(str) - 1;  while (i < j) { //swaping end characters  temp = str[i];  str[i] = str[j];  str[j] = temp;  i++;  j--;  }  }  int main() {  char str[20];  printf("Enter a string: ");  fgets(str, sizeof(str), stdin);  printf("String is %s",str);  reverse(str);  printf("Reversed string : %s\n", str);  return 0;  } | |
| Output | |
| Program No: 15 | Date: 23/07/2025 |
| Program Title : Read n Strings and display them in the ascending order. | |
| /\* PROGRAM-15 SORTING N STRINGS  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 28/07/2025  \*/  #include <stdio.h>  #include <string.h>  int main() {  int n, i, j;  char str[20][100], temp[100];  printf("Enter the number of strings: ");  scanf("%d", &n);  getchar(); // to consume the newline after scanf  for (i = 0; i < n; i++) {  printf("Enter string %d: ", i + 1);  fgets(str[i], sizeof(str[i]), stdin);  str[i][strcspn(str[i], "\n")] = '\0'; // remove newline  }  for (i = 0; i < n - 1; i++) { // Sort strings using bubble sort  for (j = i + 1; j < n; j++) {  if (strcmp(str[i], str[j]) > 0) {  strcpy(temp, str[i]);  strcpy(str[i], str[j]);  strcpy(str[j], temp);  }  }  }  printf("\nStrings in ascending order:\n");  for (i = 0; i < n; i++) {  printf("%s\n", str[i]);  }  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 16 | Date: 23/07/2025 |
| Program Title : Reverse a string using Stack. | |
| /\* PROGRAM-16 STRING REVERSAL USING STACK  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 28/07/2025  \*/  #include <stdio.h>  char stack[100];  int top=-1;  void push(char e) { //function to push elements onto stack  if (top + 1 == 100) {  printf("Error: Stack is Full\n");  }  else {  stack[++top] = e;  printf("%c",e);  }  }  void pop(){ //function to top pop element from stack  if (top == -1) {  printf("Error: Stack is Empty\n");  }  else {  printf("%c", stack[top--]);  }  }  void peek() { //function to display the top element of the stack  int i;  if (top == -1) {  printf("Stack is Empty\n");  }  else {  printf("%c \n", stack[top]);  }  }  int menu() {  int ch;  printf("\nPush - 1\nPop - 2\nPeek - 3\nReverse - 4\nExit - 5\nEnter your choice: ");  scanf("%d", &ch);  return ch;  }  void processArray() {  int ch;  char value;  for (ch = menu(); ch != 5; ch = menu()) {  switch (ch) {  case 1:  printf("Enter value to insert: ");  scanf(" %c", &value);  push(value);  break;  case 2:  pop();  break;  case 3:  peek();  break;  case 4:  while (top != -1) { //function to pop all elements  pop();  }  break;  default:  printf("Error: Wrong Choice.\n");  }  }  }  int main() {  processArray();  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 17 | Date: 10/09/2025 |
| Program Title : Convert an expression from infix expression to postfix using stack. | |
| /\*Program 17 INFIX TO POSTFIX CONVERSION USING STACK  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 12/9/2025  \*/  #include <stdio.h>  #include <ctype.h>  #include <string.h>  #define s 20  char stack[s], expr[s], post[s];  int top = -1, j = 0;  void push(char e) { // Function to insert element into stack  stack[++top] = e;  }  char pop() { // Function to delete top element  if (top == -1) {  printf("Error: Stack is empty");  return '\0';  } else {  return stack[top--];  }  }  char peek() { // Function to return top element  if (top == -1) {  return '\0';  } else {  return stack[top];  }  }  int isOperator(char c) { // Function to check if character is operator  return (c == '+' || c == '-' || c == '\*' || c == '/' || c == '^');  }  char chariden(char d) { // Function to identify character type  if (isalnum(d)) {  return 'n';  } else if (isOperator(d)) {  return 'o';  } else if (d == '(' || d == ')') {  return 'p';  } else {  return 'x';  }  }  int precedence(char d) { // Function to return precedence of operators  switch (d) {  case '^': return 3;  case '\*':  case '/': return 2;  case '+':  case '-': return 1;  default: return 0;  }  }  void oper(char d) { // Function to handle operator  int e, f;  e = precedence(d);  f = precedence(peek());  while (top != -1 && isOperator(peek()) && (e <= f)) {  post[j++] = pop();  f = precedence(peek());  }  push(d);  }  void para(char d) { // Function to handle parenthesis  if (d == '(') {  push(d);  } else if (d == ')') {  while (top != -1 && peek() != '(') {  post[j++] = pop();  }  if (peek() == '(')  pop(); // Remove '(' from stack  else  printf("Error: Mismatched parentheses\n");  }  }  void IntoPost(char expr[s]) { // Function to convert infix to postfix  int i;  char c, d;  for (i = 0; expr[i] != '\0'; i++) {  d = expr[i];  c = chariden(d);  switch (c) {  case 'o':  oper(d);  break;  case 'n':  post[j++] = d;  break;  case 'p':  para(d);  break;  default:  break;  }  }  while (top != -1) {  if (peek() == '(') {  printf("Error: Mismatched parentheses\n");  pop();  } else {  post[j++] = pop();  }  }  post[j] = '\0'; // Null terminate postfix string  printf("Postfix Expression: %s\n", post);  }  int main() {  printf("Enter the expression: ");  scanf("%s", expr);  IntoPost(expr);  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 18 | Date: 10/09/2025 |
| Program Title : Convert an expression from infix expression to prefix using stack | |
| /\*Program 18 INFIX TO PREFIX CONVERSION USING STACK  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 12/9/2025  \*/  #include <stdio.h>  #include <ctype.h>  #include <string.h>  #define s 20  char stack[s], expr[s], pre[s];  int top = -1, j = 0;  void push(char e) { // Function to insert element into stack  stack[++top] = e;  }  char pop() { // Function to delete top element  if (top == -1) {  printf("Error: Stack is empty");  return '\0';  } else {  return stack[top--];  }  }  char peek() { // Function to return top element  if (top == -1) {  return '\0';  } else {  return stack[top];  }  }  int isOperator(char c) { // Function to check if character is operator  return (c == '+' || c == '-' || c == '\*' || c == '/' || c == '^');  }  char chariden(char d) { // Function to identify character type  if (isalnum(d)) {  return 'n';  } else if (isOperator(d)) {  return 'o';  } else if (d == '(' || d == ')') {  return 'p';  } else {  return 'x';  }  }  int precedence(char d) { // Function to return precedence of operators  switch (d) {  case '^': return 3;  case '\*':  case '/': return 2;  case '+':  case '-': return 1;  default: return 0;  }  }  void oper(char d) { // Function to handle operator  int e, f;  e = precedence(d);  f = precedence(peek());  while (top != -1 && isOperator(peek()) && ((e < f) || (e == f))) {  pre[j++] = pop();  }  push(d);  }  void para(char d) { // Function to handle parenthesis  if (d == '(') {  push(d);  } else if (d == ')') {  while (top != -1 && peek() != '(') {  pre[j++] = pop();  }  if (peek() == '(')  pop(); // Remove '(' from stack  else  printf("Error: Mismatched parentheses\n");  }  }  void reverse(char str[s]) { // Function to reverse the string and swap parenthesis  int i = 0, j;  char temp;  j = strlen(str) - 1;  while (i < j) { // Swapping end characters  temp = str[i];  str[i] = str[j];  str[j] = temp;  i++;  j--;  }  int len = strlen(str);  for (i = 0; i < len; i++) {  if (str[i] == '(')  str[i] = ')';  else if (str[i] == ')')  str[i] = '(';  }  }  void IntoPre(char expr[s]) { // Function to convert infix to prefix  int i;  char c, d;  for (i = 0; expr[i] != '\0'; i++) {  d = expr[i];  c = chariden(d);  switch (c) {  case 'o':  oper(d);  break;  case 'n':  pre[j++] = d;  break;  case 'p':  para(d);  break;  default:  break;  }  }  while (top != -1) {  if (peek() == '(') {  printf("Error: Mismatched parentheses\n");  pop();  } else {  pre[j++] = pop();  }  }  pre[j] = '\0'; // Null terminate prefix string  reverse(pre);  printf("Prefix Expression: %s\n", pre);  }  int main() {  printf("Enter the expression: ");  scanf("%s", expr);  reverse(expr);  IntoPre(expr);  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 19 | Date: 10/09/2025 |
| Program Title : Evaluate an infix expression using stack | |
| /\*Program 19 EVALUATION OF INFIX EXPRESSION  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 15/9/2025  \*/  #include <stdio.h>  #include <ctype.h>  int precedence(char op) { // function to return precedence of operators  switch (op) {  case '/':  case '\*': return 1;  case '+':  case '-': return 0;  }  return -1;  }  int apply(int a, int b, char sign) { // function to apply the operator on two operands  switch (sign) {  case '+': return a + b;  case '-': return a - b;  case '\*': return a \* b;  case '/': return a / b;  }  return 0;  }  int evaluate(char a[100]) { // function to evaluate infix expression  int i = 0, vstack[100], vtop = -1, otop = -1;  char opstack[100], sign;  int v1, v2;  int number = 0; // variable to store the parsed number  while (a[i] != '\0') {  if (isdigit(a[i])) {  number = 0;  while (isdigit(a[i])) {  number = number \* 10 + (a[i] - '0');  i++;  }  vstack[++vtop] = number;  continue; // Go to the next character in the outer loop  } else if (a[i] == '+' || a[i] == '-' || a[i] == '\*' || a[i] == '/') {  while (otop != -1 && precedence(opstack[otop]) >= precedence(a[i])) {  v2 = vstack[vtop--];  v1 = vstack[vtop--];  sign = opstack[otop--];  vstack[++vtop] = apply(v1, v2, sign);  }  opstack[++otop] = a[i];  }  i++;  }  while (otop != -1) {  v2 = vstack[vtop--];  v1 = vstack[vtop--];  sign = opstack[otop--];  vstack[++vtop] = apply(v1, v2, sign);  }  return vstack[0];  }  int main() {  char a[100];  int result;  printf("enter infix expression: ");  scanf("%s", a);  result = evaluate(a);  printf("RESULT: %d", result);  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 20 | Date: 10/09/2025 |
| Program Title : Evaluate an expression using stack by converting it into postfix before evaluating. | |
| /\*Program 20 EVALUATING INFIX TO POSTFIX EXPRESSION  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 15/9/2025  \*/  #include <stdio.h>  #include <ctype.h>  #include <string.h>  #define s 20  char stack[s], expr[s], post[s];  int top = -1, j = 0;  void push(char e) { stack[++top] = e; } // Function to insert element into stack  char pop() { // Function to delete top element  if (top == -1) {  printf("Error: Stack is empty");  return '\0';  } else {  return stack[top--];  }  }  char peek() { // Function to return top element  if (top == -1) {  return '\0';  } else {  return stack[top];  }  }  int isOperator(char c) { return (c == '+' || c == '-' || c == '\*' || c == '/' || c == '^'); } // Function to check if character is operator  char chariden(char d) { // Function to identify character type  if (isalnum(d)) {  return 'n';  } else if (isOperator(d)) {  return 'o';  } else if (d == '(' || d == ')') {  return 'p';  } else {  return 'x';  }  }  int precedence(char d) { // Function to return precedence of operators  switch (d) {  case '^': return 3;  case '\*':  case '/': return 2;  case '+':  case '-': return 1;  default: return 0;  }  }  void oper(char d) { // Function to handle operator  int e, f;  e = precedence(d);  f = precedence(peek());  while (top != -1 && isOperator(peek()) && (e <= f)) {  post[j++] = pop();  f = precedence(peek());  }  push(d);  }  void para(char d) { // Function to handle parenthesis  if (d == '(') {  push(d);  } else if (d == ')') {  while (top != -1 && peek() != '(') {  post[j++] = pop();  }  if (peek() == '(')  pop(); // Remove '(' from stack  else  printf("Error: Mismatched parentheses\n");  }  }  void IntoPost(char expr[s]) { // Function to convert infix to postfix  int i;  char c, d;  for (i = 0; expr[i] != '\0'; i++) {  d = expr[i];  c = chariden(d);  switch (c) {  case 'o':  oper(d);  break;  case 'n':  post[j++] = d;  break;  case 'p':  para(d);  break;  default:  break;  }  }  while (top != -1) {  if (peek() == '(') {  printf("Error: Mismatched parentheses\n");  pop();  } else {  post[j++] = pop();  }  }  post[j] = '\0'; // Null terminate postfix string  printf("postfix Expression: %s\n", post);  }  int evaluatePostfix(char post[s]) { // function to evaluate postfix expression  int i = 0, vstack[100], vtop = -1;  int v1, v2;  while (post[i] != '\0') {  if (isdigit(post[i])) {  vstack[++vtop] = post[i] - '0'; // Convert char to int  } else if (isOperator(post[i])) {  v2 = vstack[vtop--];  v1 = vstack[vtop--];  vstack[++vtop] = apply(v1, v2, post[i]);  }  i++;  }  return vstack[vtop];  }  int main() {  printf("Enter the infix expression: ");  scanf("%s", expr);  printf("The infix expression:\n%s\n", expr);  IntoPost(expr);  printf("The postfix expression:\n%s\n", post);  printf("The evaluated result is: %d\n", evaluatePostfix(post));  return 0;  }  int main() {  printf("Enter the infix expression: ");  scanf("%s", expr);  printf("The infix expression:\n%s\n", expr);  IntoPost(expr);  printf("The postfix expression:\n%s\n", post);  printf("The evaluated result is: %d\n", evaluatePostfix(post));  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 21 | Date: 10/09/2025 |
| Program Title : A letter means push and an asterisk means pop in the following sequence. Give the sequence of values returned by the pop operations when this sequence of operations is performed on an initially empty LIFO stack. | |
| /\*Program 21 STACK ASTERISK POPPING  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 17/9/2025  \*/  #include<stdio.h>  #define size 100  char expr[size];  int top=-1;  void push(char a) { //function to insert element into stack  if (top + 1 == size)  printf("Stack is Full");  else {  expr[++top] = a;  }  }  char pop() { //function to delete top element  if (top == -1) {  printf("Stack is Empty\n");  return '\0';  } else {  printf("%c ",expr[top--]);  return expr[top];  }  }  char peek() { //function to return top element  if (top == -1){  printf("Stack is Empty");  return '\0';  }  else{  printf("Top element: %c",expr[top]);  return expr[top];  }  }  void check(char e){ //function to check and pop asterisks  if (e == '\*' ){  pop();  }  else  push(e);  }  int main(){  int i;  printf("Enter the String:");  scanf("%s",expr);  for (i = 0;expr[i] != '\0'; i++){ //traversing through the string  check(expr[i]);  }  while (top != -1){ //popping remaining elements  pop();  }  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 22 | Date: /09/2025 |
| Program Title : Read and display a sparse matrix | |
| /\*Program 22 SPARSE MATRIX ADDITION  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 23/07/2025  \*/  #include<stdio.h> | |
| Output | |

|  |  |
| --- | --- |
| Program No: 01 | Date: 23/07/2025 |
| Program Title : Write programs to demonstrate the use of storage classes (local variable, global variable, static variable, register variable) in C. | |
| /\*Program 1 USE DIFFERENT STORAGE CLASSES (LOCAL,GLOBAL,STATIC,REGISTER)  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 23/07/2025  \*/  #include<stdio.h>  int a=10,b=10; //global var  void display() {  int c=10;  static int d=10; //static var  printf("local under increment: %d\t static under increment:%d\n",c++,d++);  }  int main(){  register int e=11; //register var  int b=5; //local var  printf("Global variable: %d\nLocal variable: %d\n",a,b);  printf("Register variable: %d\n",e);  printf("Static variable: \n");  display();  display();  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 01 | Date: 23/07/2025 |
| Program Title : Write programs to demonstrate the use of storage classes (local variable, global variable, static variable, register variable) in C. | |
| /\*Program 1 USE DIFFERENT STORAGE CLASSES (LOCAL,GLOBAL,STATIC,REGISTER)  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 23/07/2025  \*/  #include<stdio.h>  int a=10,b=10; //global var  void display() {  int c=10;  static int d=10; //static var  printf("local under increment: %d\t static under increment:%d\n",c++,d++);  }  int main(){  register int e=11; //register var  int b=5; //local var  printf("Global variable: %d\nLocal variable: %d\n",a,b);  printf("Register variable: %d\n",e);  printf("Static variable: \n");  display();  display();  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 01 | Date: 23/07/2025 |
| Program Title : Write programs to demonstrate the use of storage classes (local variable, global variable, static variable, register variable) in C. | |
| /\*Program 1 USE DIFFERENT STORAGE CLASSES (LOCAL,GLOBAL,STATIC,REGISTER)  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 23/07/2025  \*/  #include<stdio.h>  int a=10,b=10; //global var  void display() {  int c=10;  static int d=10; //static var  printf("local under increment: %d\t static under increment:%d\n",c++,d++);  }  int main(){  register int e=11; //register var  int b=5; //local var  printf("Global variable: %d\nLocal variable: %d\n",a,b);  printf("Register variable: %d\n",e);  printf("Static variable: \n");  display();  display();  return 0;  } | |
| Output | |

|  |  |
| --- | --- |
| Program No: 01 | Date: 23/07/2025 |
| Program Title : Write programs to demonstrate the use of storage classes (local variable, global variable, static variable, register variable) in C. | |
| /\*Program 1 USE DIFFERENT STORAGE CLASSES (LOCAL,GLOBAL,STATIC,REGISTER)  @ALBIN MAMMEN MATHEW  Roll No: 08  Date: 23/07/2025  \*/  #include<stdio.h>  int a=10,b=10; //global var  void display() {  int c=10;  static int d=10; //static var  printf("local under increment: %d\t static under increment:%d\n",c++,d++);  }  int main(){  register int e=11; //register var  int b=5; //local var  printf("Global variable: %d\nLocal variable: %d\n",a,b);  printf("Register variable: %d\n",e);  printf("Static variable: \n");  display();  display();  return 0;  } | |
| Output | |