**EX NO : 1 ARRAYS**

**02-09-2022**

**AIM :**

To implement the programs in Array in C Language and understand the concept.

**EX : 1a**

**Q1//Matrix multiplication**

#include<stdio.h>

void main()

{

int arr1[3][3] = {1, 1, 1, 1, 1, 1, 1, 1, 1};

int arr2[3][3] = {1, 1, 1, 1, 1, 1, 1, 1, 1};

int pdt[3][3];

int count, i = 0, j = 0, k = 0;

for(count = 0; count < 9; count++)

{

pdt[i][j] = arr1[i][j] \* arr2[j][k] + arr1[i][j + 1] \* arr2[j + 1][k] + arr1[i][j + 2] \* arr2[j+2][k];

printf("%d ", pdt[i][j]);

j = 0;

if(count % 3 == 2)

{

k = 0;

i += 1;

printf("\n");

}

else k += 1;

}

}

**OUTPUT :**

3 3 3

3 3 3

3 3 3

**Q2//Transpose of a Matrix**

#include<stdio.h>

void main()

{

int row, col;

printf("Enter Row and Column : ");

scanf("%d %d", &row, &col);

int i, j, arr[row][col];

printf("Enter Array Elements :\n");

for(i = 0; i < row; i++)

{

for(j = 0; j < col; j++)

{

scanf("%d", &arr[i][j]);

}

}

for(i = 0; i < col; i++)

{

for(j = 0; j < row; j++)

{

printf("%d ", arr[j][i]);

}

printf("\n");

}

}

**OUTPUT :**

Enter Row and Column : 3 3

Enter Array Elements :

1 2 3 4 5 6 7 8 9

1 4 7

2 5 8

3 6 9

**Q3//Intersection of Sets**

#include<stdio.h>

void main()

{

int arr1[6] = {1, 2, 3, 4, 5, 6};

int arr2[3] = {1, 3, 5};

int i, j;

printf("Intersection Elements : ");

for(i = 0; i < 6; i++)

{

for(j = 0; j < 3; j++)

{

if(arr1[i] == arr2[j])

{

printf("%d ", arr1[i]);

break;

}

}

}

}

**OUTPUT :**

Intersection Elements : 1 3 5

**Q4//Adding Odd Elements in an Array**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int len;

printf("Enter Array Length : ");

scanf("%d", &len);

int i;

int \*arr = (int\*)malloc(len \* sizeof(int));

printf("Enter Array Elements :\n");

for(i = 0; i < len; i++)

{

scanf("%d", arr + i);

}

int oddsum = 0;

for(i = 0; i < len; i += 2)

{

oddsum += \*(arr + i);

}

printf("OddSum = %d", oddsum);

}

**OUTPUT :**

Enter Array Length : 5

Enter Array Elements :

1 2 3 4 5

OddSum = 9

**Q5//Square Array Elements**

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

void main()

{

int len;

printf("Enter Array Length : ");

scanf("%d", &len);

int i;

int \*arr = (int\*)malloc(len \* sizeof(int));

printf("Enter Array Elements :\n");

for(i = 0; i < len; i++)

{

scanf("%d", arr + i);

}

for(i = 0; i < len; i++)

{

printf("%d ", (int)pow(\*(arr + i), 2));

}

}

**OUTPUT :**

Enter Array Length : 5

Enter Array Elements :

1 2 3 4 5

1 4 9 16 25

**Q6//Mean of Array Elements**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int len;

printf("Enter Array Length : ");

scanf("%d", &len);

int i;

int \*arr = (int\*)malloc(len \* sizeof(int));

float sum = 0;

printf("Enter Array Elements :\n");

for(i = 0; i < len; i++)

{

scanf("%d", arr + i);

sum += \*(arr + i);

}

printf("Mean = %.2f", sum/len);

}

**OUTPUT :**

Enter Array Length : 5

Enter Array Elements :

1 2 3 4 5

Mean = 3.00

**Q7//Reverse an Array**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int len;

printf("Enter Array Length : ");

scanf("%d", &len);

int i;

int \*arr = (int\*)malloc(len \* sizeof(int));

printf("Enter Array Elements :\n");

for(i = 0; i < len; i++)

{

scanf("%d", arr + i);

}

printf("Reversed Array : ");

for(i = len-1; i >= 0; i--)

{

printf("%d ", \*(arr + i));

}

}

**OUTPUT :**

Enter Array Length : 5

Enter Array Elements :

1 2 3 4 5

Reversed Array : 5 4 3 2 1

**Q8//Peak Element of an Array**

#include<stdio.h>

#include<stdlib.h>

int PeakElement(int \*arr, int len)

{

int i;

for(i = 1; i < len - 1; i++)

{

if((\*(arr + i) > \*(arr + i - 1)) && (\*(arr + i) > \*(arr + i + 1)))

return i;

}

return -1;

}

void main()

{

int len;

printf("Enter Array Length : ");

scanf("%d", &len);

int i;

int \*arr = (int\*)malloc(len \* sizeof(int));

printf("Enter Array Elements :\n");

for(i = 0; i < len; i++)

{

scanf("%d", arr + i);

}

int peak = PeakElement(arr, len);

if(peak != -1)

printf("Peak Element Index is %d \nElement is %d", peak, arr[peak]);

else

printf("None");

}

**OUTPUT :**

Enter Array Length : 5

Enter Array Elements :

1 2 3 2 1

Peak Element Index is 2

Element is 3

**Q9//Min and Max Elements in an Array**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int len;

printf("Enter Array Length : ");

scanf("%d", &len);

int i;

int \*arr = (int\*)malloc(len \* sizeof(int));

printf("Enter Array Elements :\n");

for(i = 0; i < len; i++)

{

scanf("%d", arr + i);

}

int min = \*(arr + 0), max = \*(arr + 0);

for(i = 1; i < len; i++)

{

if(min > \*(arr + i))

{

min = \*(arr + i);

continue;

}

if(max < \*(arr + i))

max = \*(arr + i);

}

printf("Min = %d \nMax = %d", min, max);

}

**OUTPUT :**

Enter Array Length : 5

Enter Array Elements :

1 2 3 4 5

Min = 1

Max = 5

**Q10//Weighted Average of a Number**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int len;

printf("Enter Array Length : ");

scanf("%d", &len);

int i;

int \*arr = (int\*)malloc(len \* sizeof(int));

printf("Enter Array Elements :\n");

for(i = 0; i < len; i++)

{

scanf("%d", arr + i);

}

float sum = 0;

for(i = 0; i < len; i++)

{

sum += \*(arr + i) \* (i + 1);

}

printf("%.2f", sum/len);

}

**OUTPUT:**

Enter Array Length : 5

Enter Array Elements :

1 2 3 4 5

11.00

**Q11//Add Two Matrices**

#include<stdio.h>

void main()

{

int row, col;

printf("Enter Row and Column : ");

scanf("%d %d", &row, &col);

int i, j, arr1[row][col], arr2[row][col], sum[row][col];

printf("Enter Array Elements : format(Arr1 Arr2)\n");

for(i = 0; i < row; i++)

{

for(j = 0; j < col; j++)

{

scanf("%d %d", &arr1[i][j], &arr2[i][j]);

}

}

printf("Sum Array :\n");

for(i = 0; i < row; i++)

{

for(j = 0; j < col; j++)

{

printf("%d ", arr1[i][j] + arr2[i][j]);

}

printf("\n");

}

}

**OUTPUT :**

Enter Row and Column : 3 3

Enter Array Elements : format(Arr1 Arr2)

1 1

2 2

3 3

4 4

5 5

6 6

7 7

8 8

9 9

Sum Array :

2 4 6

8 10 12

14 16 18

**Q12//Array Operations**

#include<stdio.h>

#include<stdlib.h>

void insert(int \*arr, int \*len, int elmt, int pos)

{

int i, temp;

\*len += 1;

for(i = pos-1; i < \*len; i++)

{

temp = \*(arr + i);

\*(arr + i) = elmt;

elmt = temp;

}

}

void del(int \*arr, int \*len, int pos)

{

int i;

\*len -= 1;

for(i = pos-1; i < \*len; i++)

{

\*(arr + i) = \*(arr + i + 1);

}

}

int search(int \*arr, int \*len, int elmt)

{

int i;

for(i = 0; i < \*len; i++)

{

if(\*(arr + i) == elmt)

{

return i;

}

}

return -1;

}

void print(int \*arr, int \*len)

{

int i;

printf("Array : ");

for(i = 0; i < \*len; i++)

{

printf("%d ", \*(arr + i));

}

}

void main()

{

int i;

int \*arr = (int\*)malloc(10 \* sizeof(int));

int len;

printf("Enter Array Length : ");

scanf("%d", &len);

printf("Enter Array Elements :\n");

for(i = 0; i < len; i++)

{

scanf("%d", arr + i);

}

int schind;

int elmt, pos;

int check = 1 ,ch;

while(check != 0)

{

printf("1 : Insert \n2 : Delete \n3 : Search\n");

scanf("%d", &ch);

switch(ch)

{

case 1:

printf("Enter Element and Position : ");

scanf("%d %d", &elmt, &pos);

insert(arr, &len, elmt, pos);

print(arr, &len);

break;

case 2:

printf("Enter Delete Element Position : ");

scanf("%d", &pos);

del(arr, &len, pos);

print(arr, &len);

break;

case 3:

printf("Enter Search Element : ");

scanf("%d", &elmt);

schind = search(arr, &len, elmt);

if(schind != -1)

{

printf("Index of %d is %d", elmt, schind);

}

else

printf("Not Found");

break;

default :

printf("Enter Valid Choice ");

}

printf("\nTo Exit Press 0 ");

scanf("%d", &check);

}

}

**OUTPUT :**

Enter Array Length : 5

Enter Array Elements :

1 2 3 4 5

1 : Insert

2 : Delete

3 : Search

2

Enter Delete Element Position : 3

Array : 1 2 4 5

To Exit Press 0 1

1 : Insert

2 : Delete

3 : Search

1

Enter Element and Position : 3 3

Array : 1 2 3 4 5

To Exit Press 0 2

1 : Insert

2 : Delete

3 : Search

3

Enter Search Element : 3

Index of 3 is 2

To Exit Press 0 0

**EX : 1b**

**Q1//Duplicate Elements in an Array**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int \*p, i, j, len, temp;

printf("Enter Array Length : ");

scanf("%d", &len);

p = (int\*)malloc(len\*sizeof(int));

printf("Enter Array Elements : ");

for(i = 0; i < len; i++)

{

scanf("%d", p + i);

}

int \*count = (int\*)calloc(len, sizeof(int)), sum = 0;

for(i = 0; i < len; i++)

{

\*(count + \*(p + i)) += 1;

sum += \*(count + \*(p + i));

}

if(sum < len + 1)

printf("%d ", -1);

else

{

for(i = 0; i < len; i++)

{

if(\*(count + i) > 1)

printf("%d ", i);

}

}

}

**OUTPUT :**

Enter Array Length : 5

Enter Array Elements : 1 2 2 3 4

2

**Q2//Missing Number and Repeating Number**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int \*p, i, j, len, temp;

printf("Enter Array Length : ");

scanf("%d", &len);

p = (int\*)malloc(len\*sizeof(int));

printf("Enter Array Elements : ");

for(i = 0; i < len; i++)

{

scanf("%d", p + i);

}

int \*count = (int\*)calloc(len, sizeof(int));

for(i = 0; i < len; i++)

{

\*(count + \*(p + i)) += 1;

}

for(i = 1; i <= len; i++)

{

if (\*(count + i) > 1) printf("\nRepeat : %d", i);

if (\*(count + i) == 0) printf("\nMiss : %d", i);

}

}

**OUTPUT :**

Enter Array Length : 5

Enter Array Elements : 1 2 2 4 5

Repeat : 2

Miss : 3

**Q3// Spiral Pattern**

#include <stdio.h>

void main()

{

int row, col, i, j;

printf("Enter Row and Column : ");

scanf("%d %d", &row, &col);

int arr[row][col];

printf("Enter Elements :\n");

for(i = 0; i < row; i++)

{

for(j = 0; j < col; j++)

{

scanf("%d", &arr[i][j]);

}

}

int k = 0, l = row-1, m = 0, n = col-1;

while(k <= l && m <= n)

{

for(i = m; i <= n; i++)

{

printf("%d ", arr[k][i]);

}

k++;

for(i = k; i <= l; i++)

{

printf("%d ", arr[i][n]);

}

n--;

for(i = n; i >= m; i--)

{

printf("%d ", arr[l][i]);

}

l--;

for(i = l; i >= k; i--)

{

printf("%d ", arr[i][m]);

}

m++;

}

}

**OUTPUT :**

Enter Row and Column : 4 4

Enter Elements :

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

**Q4//Finding kth smallest Element**

#include<stdio.h>

void sort(int \*p, int len)

{

int i, j = 1;

for(i = 0; i < len; i++)

{

if(j < len)

{

if(\*(p+i) > \*(p+j))

{

int temp = \*(p + i);

\*(p + i) = \*(p + j);

\*(p + j) = temp;

}

i--;

j++;

}

else j = i + 2;

}

}

void main()

{

int len, i;

printf("Enter Length : ");

scanf("%d", &len);

int arr[len];

printf("Enter Array Elements : ");

for(i = 0; i < len; i++)

{

scanf("%d", &arr[i]);

}

sort(arr, len);

printf("Enter K Value : ");

scanf("%d", &i);

printf("Kth Smallest Element : %d", arr[i-1]);

}

**OUTPUT :**

Enter Length : 5

Enter Array Elements : 1 10 9 7 5

Enter K Value : 3

Kth Smallest Element : 7

**Q5//Division of Sum of Pairs**

#include<stdio.h>

int pairsum(int \*p, int val, int len)

{

int i, j = 1;

for(i = 0; i < len; i++)

{

if(j < len)

{

if((\*(p+i) + \*(p+j))%val == 0) return 1;

i--;

j++;

}

else j = i + 2;

}

return 0;

}

void main()

{

int len, i;

printf("Enter Length : ");

scanf("%d", &len);

int arr[len];

printf("Enter Array Elements : ");

for(i = 0; i < len; i++)

{

scanf("%d", &arr[i]);

}

printf("Enter Divisor Value : ");

scanf("%d", &i);

if(pairsum(arr, i, len) == 1) printf("True");

else printf("False");

}

**OUTPUT :**

1)Enter Length : 5

Enter Array Elements : 2 4 3 1 6

Enter Divisor Value : 4

True

2)Enter Length : 4

Enter Array Elements : 9 5 1 10

Enter Divisor Value : 4

False

**Q6a//Sum of Array using DMA**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int len;

printf("Enter Array Length : ");

scanf("%d", &len);

int \*p = (int\*)malloc(sizeof(int) \* len);

int i;

printf("Enter Array Elements : ");

for(i = 0; i < len; i++)

{

scanf("%d", p + i);

}

printf("Array Elements : ");

int sum = 0;

for(i = 0; i < len; i++)

{

sum += \*(p + i);

printf("%d ", \*(p + i));

}

printf("\nSum = %d", sum);

}

**OUTPUT :**

Enter Array Length : 5

Enter Array Elements : 1 2 3 4 5

Array Elements : 1 2 3 4 5

Sum = 15

**Q6b//Array of Pointers**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int \*p[5];

int a = 1, b = 2, c = 3, d = 4, e = 5;

p[0] = &a;

p[1] = &b;

p[2] = &c;

p[3] = &d;

p[4] = &e;

for(int i = 0; i < 5; i++)

{

printf("Address is %d and Value is %d\n", p[i], \*p[i]);

}

}

**:**

Address is 293599804 and Value is 1

Address is 293599800 and Value is 2

Address is 293599796 and Value is 3

Address is 293599792 and Value is 4

Address is 293599788 and Value is 5

**Q7//Access of Structures**

#include<stdio.h>

#include<stdlib.h>

struct Student{

int id;

char name[30];

}s1, \*s2;

void main()

{

printf("Name : ");

scanf("%s", s1.name);

printf("ID : ");

scanf("%d", &s1.id);

s2 = &s1;

printf("Student 1 \nNormal\nID : %d\nName : %s\n", s1.id, s1.name);

printf("\nPointer \nID : %d\nName : %s\n", s2->id, s2->name);

}

**OUTPUT :**

Name : ABC

ID : 1

Student 1

Normal

ID : 1

Name : ABC

Pointer

ID : 1

Name : ABC

**Q8//Sum of n Numbers**

#include<stdio.h>

int addUpton(int n)

{

return n\*(n+1)/2;

}

void main()

{

int num;

printf("Enter Number : ");

scanf("%d", &num);

printf("Sum upto n : %d", addUpton(num));

}

**OUTPUT :**

Enter Number : 10

Sum upto n : 55

**Ex : 1c**

**Q1//Array Address Calculation**

#include<stdio.h>

#include<stdlib.h>

struct arr

{

int elmt, adr;

}\*p;

void main()

{

int len, i, search, size = 2;

printf("Enter Array Length : ");

scanf("%d", &len);

p = (struct arr\*)malloc(sizeof(struct arr)\*len);

printf("Enter Base Address and First Element : ");

scanf("%d %d", &p->adr, &p->elmt);

printf("Enter Array Elements : ");

for(i = 1; i < len; i++)

{

scanf("%d", &(p+i)->elmt);

(p+i)->adr = p->adr + size\*(i- 0);

}

printf("Enter Search Element : ");

scanf("%d", &search);

for(i = 0; i < len; i++)

{

if((p+i)->elmt == search)

{

printf("Address of %d is %d", (p+i)->elmt, (p+i)->adr);

}

}

}

**OUTPUT :**

Enter Array Length : 5

Enter Base Address and First Element : 1020 1300

Enter Array Elements : 1400 1500 1600 1700

Enter Search Element : 1600

Address of 1600 is 1026

**Q2//2-D Array Address Calculation**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int base, size, m, n, x, y, i, j;

printf("Enter Values for Base, datatype\_size, rows, columns, baserow and basecol : ");

scanf("%d %d %d %d %d %d", &base, &size, &m, &n, &x, &y);

printf("Enter Position(Row, Column) : ");

scanf("%d %d", &i, &j);

int adr = base + size\*(m\*(j-y) + (i-x));

printf("Address : %d", adr);

}

**OUTPUT :**

Enter Values for Base, datatype\_size, rows, columns, baserow and basecol : 100 1 10 15 1 1

Enter Position(Row, Column) : 8 6

Address : 157

**Q3//3-D Array Address Calculation**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int base, size, m, n, p, x, y, z, i, j, k;

printf("Enter Values for Base, datatype\_size, rows, columns, width, baserow, basecol and basewidth :\n");

scanf("%d %d %d %d %d %d %d %d", &base, &size, &m, &n, &p, &x, &y, &z);

printf("Enter Position(Row, Column, Width) : ");

scanf("%d %d %d", &i, &j, &k);

int adr = base + size\*(m\*n\*(i-x) + n\*(j-y) + (k-z));

printf("Address : %d", adr);

}

**OUTPUT :**

Enter Values for Base, datatype\_size, rows, columns, width, baserow, basecol and basewidth :

400 2 9 1 10 1 -4 5

Enter Position(Row, Column, Width) : 5 -1 8

Address : 484

**Q4//Malloc vs Calloc**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int len;

printf("Enter Array Length : ");

scanf("%d", &len);

int i;

int \*pm = (int\*)malloc(sizeof(int)\*len);

int \*pc = (int\*)calloc(sizeof(int),len);

printf("Malloc Array : ");

for(i = 0; i < len; i++)

{

printf("%d ", pm[i]);

}

printf("\nCalloc Array : ");

for(i = 0; i < len; i++)

{

printf("%d ", pc[i]);

}

}

**OUTPUT :**

Enter Array Length : 5

Malloc Array : 1276926816 381 1276903760 381 2035506291

Calloc Array : 0 0 0 0 0

**Q5//Free Function in C**

#include<stdio.h>

#include<stdlib.h>

void main()

{

int i;

int \*p = (int\*)calloc(sizeof(int),3);

p[0] = 1;

p[1] = 2;

p[2] = 3;

printf("Before Free : ");

for(i = 0; i < 3; i++)

{

printf("%d ", p[i]);

}

free(p);

printf("\nAfter Free : ");

for(i = 0; i < 3; i++)

{

printf("%d ", p[i]);

}

}

**OUTPUT :**

Before Free : 1 2 3

After Free : 961305440 351 961282384

**RESULT :**

Thus, the programs are executed and implemented in C.

21I205

Anand Narayanan N

Btech IT G1