**Algorithm for Select sort:**

Let ARR is an array having N elements

1. Read ARR

2. Repeat step 3 to 6 for I=0 to N-1

3. Set MIN=ARR[I] and Set LOC=I

4. Repeat step 5 for J=I+1 to N

5. If MIN>ARR[J], then

  (a) Set MIN=ARR[J]

  (b) Set LOC=J

  [End of if]

  [End of step 4 loop]

6. Interchange ARR[I] and ARR[LOC] using temporary variable

 [End of step 2 outer  
loop]

7. Exit

**Select sort Program:**

|  |
| --- |
| #include<iostream>    using namespace std;    int main()  {       int i,j,n,loc,temp,min,a[30];       cout<<"Enter the number of elements:";       cin>>n;       cout<<"\nEnter the elements\n";    for(i=0;i<n;i++)   {          cin>>a[i];       }    for(i=0;i<n-1;i++)      {       min=a[i];       loc=i;          for(j=i+1;j<n;j++)          {           if(min>a[j])           {           min=a[j];           loc=j;              }          }   temp=a[i];          a[i]=a[loc];          a[loc]=temp;      }    cout<<"\nSorted list is as follows\n";      for(i=0;i<n;i++)      {          cout<<a[i]<<" ";      }   return 0;  } |

**Program for Selection sort using recursion**

#include <iostream>

using namespace std;

int minIndex(int a[], int i, int j)

{

if (i == j)

return i;

int k = minIndex(a, i + 1, j);

return (a[i] < a[k])? i : k;

}

void recurSelectionSort(int a[], int n, int index = 0)

{

if (index == n)

return;

int k = minIndex(a, index, n-1);

if (k != index)

swap(a[k], a[index]);

recurSelectionSort(a, n, index + 1);

}

int main()

{

int arr[] = {3, 1, 5, 2, 7, 0};

int n = sizeof(arr)/sizeof(arr[0]);

recurSelectionSort(arr, n);

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

cout << arr[i] << " ";

cout << endl;

return 0;

}

**Program for Selection sort using recursion and linked list:**

#include <iostream>

using namespace std;

struct Node {

int data;

struct Node\* next;

};

void swapNodes(struct Node\*\* head\_ref, struct Node\* currX,

struct Node\* currY, struct Node\* prevY)

{

\*head\_ref = currY;

prevY->next = currX;

struct Node\* temp = currY->next;

currY->next = currX->next;

currX->next = temp;

}

struct Node\* recurSelectionSort(struct Node\* head)

{

if (head->next == NULL)

return head;

struct Node\* min = head;

struct Node\* beforeMin = NULL;

struct Node\* ptr;

for (ptr = head; ptr->next != NULL; ptr = ptr->next) {

if (ptr->next->data < min->data) {

min = ptr->next;

beforeMin = ptr;

}

}

if (min != head)

swapNodes(&head, head, min, beforeMin);

head->next = recurSelectionSort(head->next);

return head;

}

void sort(struct Node\*\* head\_ref)

{

if ((\*head\_ref) == NULL)

return;

\*head\_ref = recurSelectionSort(\*head\_ref);

}

void push(struct Node\*\* head\_ref, int new\_data)

{

struct Node\* new\_node =

(struct Node\*)malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void printList(struct Node\* head)

{

while (head != NULL) {

cout << head->data << " ";

head = head->next;

}

}

int main()

{

struct Node\* head = NULL;

push(&head, 6);

push(&head, 4);

push(&head, 8);

push(&head, 12);

push(&head, 10);

cout << "Linked list before sorting:n";

printList(head);

sort(&head);

cout << "\nLinked list after sorting:n";

printList(head);

return 0;

}