**Bubble Sort algorithm**

#include <stdio.h>

void main()

{

  int A[10], i, j, n, t, cmpcount = 0, swapcount = 0, outerloopcount = 0, innerloopcount = 0, swapcheck = 0;

  printf("key in the element to be sorted [ Maximim 10 ] \n");

  scanf("%d", &n);

  printf("Key in the %d elements \n", n);

  for (i = 0; i < n; i++)

    scanf("%d", &A[i]);

  printf("\nThe keyed in the elements are \n");

  for (i = 0; i < n; i++)

    printf("%d\t", A[i]);

  printf("\n");

  // sorting

  for (i = 0; i <= n - 2; i++)

  {

    if (i != 0 && swapcheck == 0)

      break;

    outerloopcount++;

    for (j = 0; j <= n - i - 2; j++)

    {

      innerloopcount++;

      cmpcount++;

      swapcheck = 0;

      if (A[j] > A[j + 1])

      {

        swapcheck = 1;

        swapcount++;

        t = A[j];

        A[j] = A[j + 1];

        A[j + 1] = t;

      }

    }

  }

  printf("\nThe sorted list is \n");

  for (i = 0; i <= n - 1; i++)

    printf("\t%d", A[i]);

  printf("\n");

  printf("\n\tThere were %d innerloop interations, %d outerloop iterations, %d comparisons and %d swaps. \n\n", innerloopcount, outerloopcount, cmpcount, swapcount);

}

**Selection sort algorithm**

#include <stdio.h>

void swap(int \*f, int \*s)

{

    int t = \*f;

    \*f = \*s;

    \*s = t;

}

void main()

{

    int a[10], i, j, n, cmpcount = 0, swapcount = 0, outerloopcount = 0, innerloopscount = 0, minpos;

    printf("Key in the count of elements to be sorted  [Maximum 10]\n");

    scanf("%d", &n);

    printf("Key in the %d elements\n", n);

    for (i = 0; i < n; i++)

        scanf("%d", &a[i]);

    printf("\n The keyed in elements are \n");

    for (i = 0; i < n; i++)

        printf("%d\t", a[i]);

    // sorting

    for (i = 0; i < n - 1; i++)

    {

        outerloopcount++;

        minpos = i;

        for (j = i + 1; j < n; j++)

        {

            innerloopscount++;

            cmpcount++;

            if (a[j] < a[minpos])

            {

                minpos = j;

            }

        }

        if (minpos != i)

        {

            swapcount++;

            swap(&a[minpos], &a[i]);

        }

    }

    printf("\nThe sorted list is \n");

    for (i = 0; i < n; i++)

        printf("\t%d", a[i]);

    printf("\n");

    printf("\n\tThere were %d innerloop interations, %d outerloop iterations, %d comparisons and %d swaps. \n\n", innerloopscount, outerloopcount, cmpcount, swapcount);

}

**Fractional Knapsack Algorithm**

#include<stdio.h>

struct item{

    int id;

    int w;

    int p;

    float value;

};

void main()

{

    int i,j,w,p,tw=0,capacity,itemcount;

    float tv=0,partp;

    struct item K[50];

    printf("\nKey in count of items[Maximum 50] and the maximum capacity of the bag\n\t");

    scanf("%d%d",&itemcount,&capacity);

    printf("Key in,row-wise[oneline per item],.the serial number,the weight and the profit for each of the %d items\n",itemcount);

    for(i=0;i<itemcount;i++)

    scanf("%d%d%d",&K[i].id,&K[i].w,&K[i].p);

    for(i=0;i<itemcount;i++)

    K[i].value=(float)K[i].p/K[i].w;

    struct item KK;

    for (i=0;i<itemcount-1;i++)

    {

        for(j=0;j<itemcount-1;j++)

        {

            if(K[j+1].value>K[j].value)

            {

                KK=K[j+1];

                K[j+1]=K[j];

                K[j]=KK;

            }

        }

    }

    printf("The %d items arranged in non-decendeing orders of the ratio value =[profit/weight]is as under\n",itemcount);

    printf("\n\tvalue\titem serial number\tweight\tprofit\n");

    for(i=0;i<itemcount;i++)

    printf("\n\t%0.2f\t\t%d\t\t%d\t%d\n",K[i].value,K[i].id,K[i].w,K[i].p);

    printf("\n");

    printf("The solutions to the fractional Knapsack problem \n\n");

    for(i=0;i<itemcount;i++)

    {

        if(K[i].w+tw<=capacity)

        {

            tw+=K[i].w;

            tv+=K[i].p;

            printf("\nSelected item%d [whole]\t\tweight%d\tprofit%d\t\tCumulative weight\t%d\tCumulative value\t%0.2f\n\n",K[i].id,K[i].w,K[i].p,tw,tv);

        }

        else

        {

            w=capacity-tw;

            partp=(float)w\*(float)K[i].p/(float)K[i].w;

            tw+=w;

            tv+=partp;

            printf("\nSelected items%d [part]\t\tweight%d\tprofit%0.2f\t\tCumulative weight\t\t%d\tCumulative value\t%0.2f\n\n",K[i].id,w,partp,tw,tv);

            break;

        }

    }

    printf("\nThus the Knapsack with a capacity of %d can hold items worth a Cumulative total value of \t%0.2f\n\n",tw,tv);

    printf("\n");

}