Apr 02, 07 14:43 **exo1.c** Page 1/1

```c
#include <stdio.h>
#include <stdlib.h>

/* fonction serie de l'énoncé 1 */
double serie_1(int n, double x)
{
  double u = 1.0;
  double s = u;;
  for (int i= 1; i<= n; i++)
    {
      u = u * x / i;
      s += u;
    }
  return s;
}


/* fonction serie de l'énoncé 2 */
double serie_2(int n, double x)
{
  double u = 2.0;
  double s = u;;
  for (int i= 1; i<= n; i++)
    {
      u = 3 * u * x / i;
      s += u;
    }
  return s;
}

/* la fonction main n'est pas demandé dans l'énoncé du DS */
int main(int argc, char *argv[]) {
  printf("%f\n", serie_1 (atoi(argv[1]), atof(argv[2])));
  printf("%f\n", serie_2 (atoi(argv[1]), atof(argv[2])));
}
```

Apr 02, 07 18:34 **exo2.c** Page 1/1

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

/* f supprime le dernier caractère de la chaine s */
void
f(char s[])
{
  int l= strlen(s);
  if (l != 0)
    s[l-1]='\0';
}

/* g supprime le premier caractère de la chaine  s*/
void
g(char s[])
{
  for(int i=0; s[i] != '\0'; i++)
    s[i]= s[i+1];
}

void
supprimer_k_lettres(char s[], int k)
{
  int l = strlen(s);
  int i;
  for(i=0; i+k < l; i++)
    s[i]= s[i+k];
  s[i]= '\0';
}

int
main(void)
{
  char s[]="totototo";
  f(s);
  printf("<%s>\n", s);      /*<tototot>*/
  g(s);
  printf("<%s>\n", s);      /*<ototot> */

  supprimer_k_lettres(s, 4);
  printf("<%s>\n", s);      /*<ot>     */
  supprimer_k_lettres(s, strlen(s)+4);
  printf("<%s>\n", s);      /*<>       */
  return EXIT_SUCCESS;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double
distance(double x[], double y[], int n, int i, int j)
{
  double a= x[i] − x[j];
  double b= y[i] − y[j];

  return sqrt( a*a + b*b );
}

double
distance_min_point(double x[], double y[], int n, int i)
{
  int k = (i==0)?1:0;
  double d_min= distance(x, y, n, k, i);
  for(int j= 0; j<n; j++)
    {
      if (j!= i && j!= k)
        {
          double d= distance(x, y, n, j, i);
          if( d< d_min )
            d_min= d;
        }
    }
  return d_min;
}

double
distance_min(double x[], double y[], int n)
{
  double d_min= distance(x, y, n, 0, 1);
  for(int i= 0; i < n; i++)
    for(int j=i+1; j < n; j++)
    {
      double d= distance(x, y, n, i, j);
      if( d< d_min )
        d_min= d;
    }
  return d_min;
}

static void
usage(char* s)
{
  printf("Usage: %s abscisse1 ordonnee1 abscisse2 ordonnee2 [...] \n", s);
  exit(EXIT_FAILURE);
}

void
afficher_points(double x[], double y[], int n)
{
  for(int i=0; i<n; i++)
    printf("P_%d=( %4.2f, %4.2f ) \n", i, x[i], y[i]);
}
int
main(int argc, char* argv[])
{
  if(argc < 5 || argc % 2 == 0)
    usage(argv[0]);

  int n= (argc−1)/2;
  double x[n], y[n];
  for(int i=0, j=1; i< n; i++, j+=2)
    {
      x[i]= atof(argv[j]);
      y[i]= atof(argv[j+1]);
```



```c
    }
  afficher_points(x, y , n);
  printf("Distance minimale entre une paire de points arbitraires est = %4.2f \n ",
         distance_min(x, y, n));
  return EXIT_SUCCESS;
}
```