Programa

Un lenguaje de programación nos facilita la comunicación con el ordenador para programar algoritmos matemáticos.

El lenguaje debe ser no ambiguo para programar algoritmos matemáticos.

Las instrucciones se interpretan de manera secuencial.

A continuación se presenta una introducción al lenguaje c++. Se explicarán únicamente las partes del lenguaje que hacen falta para este curso de Cálculo numérico.

Para compilar, es decir traducir del lenguaje programa a código máquina, se utiliza el comando:

g++ nombre-fichero-fuente -o nombre-fichero-ejecutable

donde nuestro programa estará en el fichero llamado nombre-fichero-fuente y el resultado de la compilación se guardará en el fichero llamado nombre-fichero-ejecutable

Para ejecutar el programa escribiremos el nombre del fichero ejecutable, precedido de "./" para indicar que está en el directorio actual. En nuestro ejemplo seria:

./nombre-fichero-ejecutable

Para abortar la ejecución de un programa se puede utilizar la secuencia de teclas "ctrl" y "c".

C++.

- Comentarios generales

  Se distinguen letras mayúsculas de letras minúsculas.

  El programa se escribirá usando un editor de texto (como por ejemplo emacs) incluyendo cabeceras al principio para uso de ciertos métodos pre-programados. Los entornos de sentencias se enmarcan entre llaves. Las sentencias se separan entre sí por punto y coma ";".

- Tipos de programa: principal, función.

  Las sentencias de un programa se enmarcan entre llaves.

  1. Programa principal

     Es la unidad basica ejecutable, que puede hacer uso de funciones. El programa principal es así mismo una función llamada "main".

     int main(){ }

  2. Función

     Es una variable cuyo valor es el resultado de un cierto algoritmo.

     Ejemplo:

     double norvec(double *v, int n){ return maxvec(s,n); }

     El valor "void" quiere decir no devuelve ningun valor en el nombre de la función (sería como la subrutina en Fortran).

     Ejemplo:

     void sumv(double *v1,double *v2,int n,double *u){ }

- Variables. Nombres simbólicos que damos a porciones de memoria donde se almacenarán datos de interés para nuestro algoritmo. Estos nombres deben comenzar siempre por una letra y pueden contener números y el guión bajo. Se distinguen mayúsculas de minúsculas. El valor de una variable es el dato que haya en la porción de memoria que representa. Las declaraciones de las variables pueden ir en cualquier parte del programa y afectan al dominio (entre llaves) en el que estén.

  Ejemplos:

  1. int contador,i,acumula1
  2. float x,y,z,norma,modulo,x1te200a
  3. long k,n,m
  4. short mcorto
  5. float normasup,modulo2,A[3][6],V[4]
  6. double suma,producto
  7. char frase[80]
  8. char *apellido
     (puntero)
  9. string prueba
     (necesita incluir string)
  10. bool test
  11. const int n=3

- Entrada-Salida con teclado y monitor.

  - std::cin

    Ejemplo:

    int canal; double energia;

    std::cin>>canal>>energia;

    Se obtendrán del teclado los valores de las variables canal y energia. Estos valores quedarán almacenados en las zonas de memoria asignadas a dichas variables.

  - std::cout

    Ejemplo:

    double Ek,V;

    std::cout<<"Energia cinetica = "<<Ek<<" Energia potencial = "<<V<<std::endl;

    Se escribirá en el monitor la frase indicada, donde se sustituyen las variables por su valor o contenido.

  - Comentarios

    1. Se necesita incluir la cabecera iostream para poder utilizar cin o cout.
       #include <iostream>

2. using namespace std;
   Nos ahorra estar escribiendo std:: delante de las clases cin, cout, endl, etc.
3. Se verá más información cuando hablemos de entrada salida en general.

- Operador de asignación: '='

  Ejemplos:

  1. k = 6
     a la variable k se le asigna el valor entero 6. Este valor quedará almacenado en la zona de memoria de 4 Bytes asociada a la variable entera k.
  2. m = k
     el valor de k se asigna a la variable m. Después de esta asignación las dos variables m y k contendrán el mismo valor.

- Valores o constantes. Son los valores que puede tomar una variable.

  Ejemplos:

  1. producto = 2.4
  2. modulo2 = 0.678E+03
  3. apellido = "Esto es una prueba"
  4. prueba = "Hola que tal"
  5. contador = -207
  6. test = true
  7. test = false

- Operaciones aritméticas: +,-,∗,/

  Ejemplos:

  1. modulo2 = x∗x + y∗y + z∗z
  2. normaliza = x/suma

- Operador potencia: *pow*

  Ejemplos:

  1. modulo2 = *pow*(x,y)
  2. cuenta = $pow(x, 3)$

- Relaciones: > (Mayor que), >= (Mayor o igual que), < (Menor que), <= (Menor o igual que), == (Igual a), ! = (Distinto de)

  Ejemplos:

1. i<= 7
2. contador==n

- Operadores lógicos: ! (No), && (Y), || (O)

  Ejemplos:

  1. (contador==n)&&(i<=7)
  2. (! test)||(k>20)

- Estructura repetitiva o bucle. Hay 3 maneras de construirla:

  1. Con contador

     for(*variable = valor-inicial; variable <= valor-final; variable=variable+incremento*){ }

     Ejemplos:

     (a) for(i=1;i<=6;i=i+1){
     cout<<pow(10,i)<<"\n";
     }

     El bucle anterior escribe las potencias
     10**1, 10**2, 10**3, 10**4, 10**5, 10**6

     (b) for(i=1;i<=6;i++){
     cout<<pow(10,i)<<"\n";
     }

     Hace lo mismo que el bucle anterior.

     (c) for(i=1;i<=6;i=i+2){
     cout<<pow(10,i)<<"\n";
     }

     Escribe las potencias
     10**1, 10**3, 10**5

  2. Con condición lógica al principio del bucle.

     while(*condicion-logica*){ }

     Ejemplos:

     (a) i = 1;
     while(i<=6){
     cout<<pow(10,i)<<"\n";
     i++;
     }

     Escribe las potencias
     10**1, 10**2, 10**3, 10**4, 10**5, 10**6

     (b) i = 1;
     while(i<= 6){
     cout<<pow(10,i)<<"\n";

i=i+2;
}

Escribe las potencias
10**1, 10**3, 10**5

3. Con condición lógica al final del bucle.
   do{ } while(*condicion − logica*);
   Ejemplos:

   (a) i=1;
   do {
   cout<<pow(10,i)<<"\n";
   i++;} while(i<=6);

   Escribe las potencias
   10**1, 10**2, 10**3, 10**4, 10**5, 10**6

- Estructura condicional

  if (*condición*) {

  } else if (*condición*) {

  } else if (*condición*) {

  } else {

  }

  Ejemplos:

  1. if(x>y){
     std::cout<<x<<"\n";
     } else {
     std::cout<<y<<"\n";
     }
  2. if (i==1){
     x = 1.2;
     y = 1.3;
     z = 2.5;
     modulo2 = x * x + y * y + z * z;
     std::cout<<"Modulo Cuadrado = "<<modulo2<<"\n";
     } else if (i==2){
     x = 10.2;
     y = 10.3;
     z = 20.5;
     modulo2 = x * x + y * y + z * z;
     std::cout<<"Modulo Cuadrado = "<<modulo2<<"\n";
     } else if (i==3) {
     x = 0.2;
     y = 0.3;
     z = 1.5;
     modulo2 = x * x + y * y + z * z;
     std::cout<<"Modulo Cuadrado = "<<modulo2<<"\n";
     } else{
     std::cout<<"Error. Este caso no existe."<<"\n";
     }

- Funciones definidas en <cmath>

  1. u=fabs(x)
  2. u=cos(x)
  3. u=acos(x)

4. u=exp(x)
5. u=sqrt(x)
6. u=sin(x)
7. u=asin(x)
8. u=tan(x)
9. u=atan(x)
10. u=log(x)
11. u=log10(x)
12. u=cosh(x)
13. u=sinh(x)
14. u=tanh(x)

- Llamada a funciones desde un programa principal.

  1. *nombre-funcion(argumentos)*
  2. *variable = nombre-funcion(argumentos)*
     El nombre de la funcion es a su vez una variable. La operación anterior asigna el valor de la funcion a otra varibale.
     Los argumentos se separan por comas.

  Ejemplos:

  1. #include <iostream>
     #include <cmath>
     #include <string>
     using namespace std;
     int main{
     const int n=2;
     double v1[n],v2[n],uu[n];
     void sumv(double *,double * ,int ,double *);
     v1[0]=1;
     v1[1]=2;
     v2[0]=3;
     v2[1]=4;
     sumv(v1,v2 ,n ,uu);
     cout<<uu[0]<<" "<<uu[1]<<"\n";
     }
     void sumv(double *v1,double * v2,int n,double *u){
     for (int i=0;i<n;i++){
     u[i]=v1[i]+v2[i];
     }
     }