Tutoría 8
Física Computacional I
Grado en Física

Javier Carrasco Serrano, javcarrasco@madrid.uned.es

Física Computacional I, Las Tablas

Tema 11:
El lenguaje C mediante ejemplos

Adobe Acrobat Document

Adobe Acrobat Document

11

10.2. Compilación, enlazado y ejecución de programas

Proyecto en Code::Blocks. Se utiliza un proyecto para dividir un programa grande en pequeños programas (archivos).

- File → New → Project → Console application → C → “HolaMundo” → “HolaMundo.cpb” (una carpeta por proyecto).
- Al elegir *console application* crea una estructura por defecto.
- File → Save file as… → “helloworld0.c” → sobre el panel de la izquierda (*workspace*), sobre “main.c” → remove file from Project → add files → “helloworld0.c”
- Añadimos *system(“PAUSE”)* al programa para que no se cierre al ejecutarse desde consola (o al abrir el .exe, que es lo mismo). Este comando deja abierta la consola hasta que se pulse alguna tecla. *stdlib.h* hace que se cargue esta función.
- Dos posiblidades:
  - Build → Compile current file → crea un objeto (O) del módulo, y ejecuta sólo esta parte del programa; rápido para detectar errores.
  - Build → Build → crea un objeto (O) por cada módulo, ejecuta todo el programa y crea el ejecutable.
  - Build and run, Run, o doble click sobre el .exe que se ha generado para correr el programa.

11.8. Punteros

Un puntero es una variable que contiene una dirección de memoria. Por tanto, se va a poder acceder a una variable a través de su dirección en memoria (*), e igualmente se va a poder operar con la dirección de memoria de una variable (&).

- * → contenido en memoria de una dirección.
- & → dirección en memoria de la variable.
- int *q → q es un puntero a un valor int, o lo que hay en q es una variable tipo int.
- *(&var)==var; &(*q)==q;

“Para el propósito de este curso y el nivel de programación que se espera conseguir, el uso de punteros no es en absoluto necesario. En muchos casos pueden ser sustituidos por otros elementos del lenguaje C como los vectores o matrices. En otros casos basta con cambiar el modo de almacenamiento de la variable en cuestión. Sin embargo, su uso permite explotar aún más a potencialidad del C y conseguir diseños de programa mucho más elegantes.”

Ejemplo: Listado 11.8

11.8. Punteros

Los vectores elementos de vectores y matrices se almacenan consecutivamente en memoria (las matrices, por filas) → nos podemos desplazar por un vector o matriz con un puntero.

Al declarar un vector, int v[5], estamos creando un puntero v que apunta a v[0], es decir, v==&v[0], y el resto de valores se guardan en las casillas de memoria v+1, … , v+4.

En general: v+n=&v[n]; *(v+n)=v[n];

Ejemplo: Listado 11.9

Una matriz es un vector de vectores, por lo que:

M[i,j] = *(*(M+i)+j) = *(M+i)[j] = *(M[i]+j)

Ejemplo: Listado 11.10

Si pensamos en las cadenas de caracteres como vectores de caracteres, también les podemos dar un tratamiento con punteros.

11.9. Lectura y escritura de datos

- *echo* (escrito desde el terminal) → escribe en el terminal (cmd) los argumentos que se le pasan por la línea de comandos.
- | (alt gr + 1) es una tubería → sirve para pasarle un input a un programa. Por ejemplo, el programa del Listado 11.6, espera que se le pase un número 0, 1, 2, 3, u otro input, y en función de ese input hace una cosa u otra. Con echo 0 | ./scanmenu (si el programa se ha guardado con ese nombre), estamos pasando un 0 como input de ese programa.
- *fopen* → abre un archivo C y devuelve un puntero a un dato tipo *FILE* (nombre archivo, estado, posición en memoria para lectura y escritura).
- Con *stdio.h* se está leyendo y escribiendo de manera transparente en *stdin / stdout*.
- *printf / scanf* → escribe/lee por defecto en *stdout / stdin.*
- *fprintf / fscanf* → escribe/lee donde se le indica (en un dato tipo *FILE*).

Ejemplo: Listado 11.11

Ejercicio 11.12

11.10. Definición de funciones: paso de argumentos por valor y por referencia

- Declaración de funciones: tipo de la función (del valor de retorno) y de sus argumentos (inputs).
- Declaración de funciones que son llamadas (utilizadas) en otras funciones: *static inline* → para funciones pequeñas, se optimiza su ejecución.

Ejemplo: Listado 11.12

- Direcciones de memoria (punteros) como entrada y salida de una función.
- Argumentos por referencia → argumentos pasados con dirección de memoria (el puntero referencia las variables) → *&variable* ; se accede al valor de la variable dentro de la función con **variable*
- Argumentos por valor → como hacíamos hasta ahora.

Ejemplo: Listado 11.13

- Los nombres de las funciones también son punteros → permite pasar una función como argumento de otra función.
- void → cuando no hay tipo ni valor de retorno, o no se sabe el tipo.

Ejemplo: Listado 11.14

11.11. Utilidad: guardar datos como una imagen

- Mapa de píxeles → matriz de números que indican cómo de cerca está cada píxel del negro / blanco.
- Hay que indicar la dimensión de la imagen, la escala (qué numero equivale al blanco; el 0 equivale al negro), y el formato.
- Permite pintar gráficas sin necesidad de utilizar un paquete externo (como Gnuplot).
- "retipado" → *cast*, cuando se trata a una variable con un tipo diferente al suyo; se indica entre paréntesis antes de la variable el tipo con el que quiere ser tratada.

Ejemplo: Listado 11.15

11.12. Estructuras de datos y definición de tipos

- *struct* → estructura de datos, conjunto de datos que el programa debe tratar de manera conjunta. Similares a objetos y sus parámetros (Java).
- Se utiliza el punto (.) para acceder a los campos de las estructuras de datos.

```
struct Vector2d {
    double x, y;
};
```

```
struct Vector2d u, v;
 ...
u.x=1.0; u.y=2.0;
v.x=2*u.x; v.y=2*u.y;
 ...
```

- *typedef* → se pueden guardar las estructuras de datos como un nuevo tipo.
- -> → para acceder a los campos de una variable (estructura) pasada por puntero.

Ejemplo: Listado 11.16

Tema 12: Métodos Monte Carlo

Adobe Acrobat Document

12

12.1. Un poco de historia

- PRÁCTICA DE C INCLUYE MONTE-CARLO. Casino de Monte-Carlo (Mónaco).
- “To explore it by experimental trial and error would have been expensive, time-consuming and hazardous. On the other hand, the problem seemed beyond the reach of theoretical calculations”.
- ¿Qué fracción de la energía de la radiación es absorbida por el medio? Radiación ionizante → daño reversible o irreversible en un cuerpo.
- Modelización de procesos aleatorios: es imposible predecir el comportamiento de una partícula individual.
- Se puede modelizar si se conoce el comportamiento promedio de un gran número y la probabilidad de los sucesos.
- Simulaciones mediante números aleatorios (generados, por ejemplo, con la ruleta de un casino).
- Diferentes sucesos con probabilidades de ocurrencia conocidas.
- Monte-Carlo → generación de sucesiones de números aleatorios. No resuelve una ecuación, simula eventos individuales y calcula promedios.
- Estimaciones con intervalos de confianza. Se reduce el error incrementando las simulaciones.
- Robustez resultado VS complejidad computacional.
- Ejemplo tirar daros a una diana para simular el valor del número pi.

12.2. Números aleatorios

- Procesos aleatorios: dado, ruleta, bingo… sucesos en los que cada evento es equiprobable, pero impredecible (distribución uniforme). Las probabilidades suman 1.
- Variable aleatoria discreta que toma valores de su espacio muestral (finito).
- Método congruente lineal: genera números “aleatorios” uniformemente distribuidos en [0,1):

$$X_{n+1} = (AX_n + B)\ mod(C)$$

X_0 es la semilla (valor inicial entero), mod es el resto de la división entera.

Se genera en cada paso un entero entre 0 y C-1 → dividiendo entre C → [0, 1).

Como los números que van saliendo están determinados por los valores de la semilla y de la ecuación lineal, en ocasiones se saca un determinado número de “aleatorios” hasta que se elige uno de ellos como nueva semilla.

Ejemplo: Listado 12.1

→ útil para la práctica de C.

→ 1234567891LL quiere decir que es un *long long int.*

12.3. Números aleatorios continuos

- Paso de variables discretas a continuas → probabilidades asociadas a intervalos, cuando la longitud de los intervalos tiende a cero → integrales.
- Ejemplo: distribución normal (o gaussiana).
- Teorema Central de Límite. $Y = X_1 + \cdots + X_n$, X_i variables aleatorias, $N > 30$ para que la estimación de la varianza sea robusta.
- $p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$
- $N(0,1)$

Ejemplo: Listado 12.2

Ojo: sqrt → indicación en la sección 10.2 del tema 10.

- Histograma → distribución gráfica de la frecuencia de un evento por intervalos. Muy útil en exploración de datos.

Ejemplo: Listado 12.3

12.4. Caminantes aleatorios y difusión browniana

- Proceso de difusión: desplazamiento de masa o energía de una región/cuerpo en el que hay más a otra en el que hay menos.
- Paseo aleatorio: movimiento sin dirección definida → "acaba" recorriendo todo el espacio (puede que en tiempo infinito…) → movimiento *browniano*. Muchas partículas → choques → difusión *browniana*.
- Suma de desplazamientos aleatorios → distribución normal de las partículas alrededor del punto de partida, D coeficiente de difusión:

$$p(x,t) = \frac{1}{\sqrt{4\pi Dt}} \exp\left(-\frac{x^2}{4Dt}\right)$$

- Difusión aleatoria en una dimensión → caminante aleatorio: avanza o retrocede un paso, 0.5 de probabilidad → distribución normal.
- Biblioteca *libprobabilidad*. Una vez construida, se pueden utilizar las funciones tantas veces como se quiera sin necesidad de volver a definirlas ☺
- La directiva *#ifndef* comprueba primero si una librería está definida (en realidad, el símbolo), y si no está, la procesa (la carga).

Listado 12.4, 12.5

12.4. Caminantes aleatorios y difusión browniana

- Compilación biblioteca *libprobabilidad*:

  gcc –c –o libprobabilidad.o libprobabilidad.c

- Cada vez que se use en un programa hay que incluir:

  #include libprobabilidad.h → en el código del programa

```
gcc -o browniano1d -lm libprobabilidad.o browniano1d.c
```

ejecutable | biblio. matemat. | biblio. probabilidad | programa fuente C

→ en el terminal.

- ¿Cómo simular un aleatorio de dos valores posibles (cara/cruz, 0/1,…)?
  - → Variable aleatoria con distribución uniforme en [0,1)
  - → Un valor si el resultado es < 0.5, el otro si es ≥ 0.5
  - → **PRIMER PASO DE LA PRÁCTICA C**
  - → **Resuelto en Listado 12.6**
  - → A partir de una distribución “aleatoria” (pseudoaleatoria), se está generando una distribución aleatoria.

12.5. Integración Monte Carlo

- Se pueden utilizar generadores de números aleatorios para aproximar problemas numéricos no aleatorios difíciles de resolver.
- Cálculo de integrales: en lugar de integrar sobre todo el dominio de integración, se “integra” (suma ponderada de valores) sobre una muestra de puntos en el dominio. Cuando aumentamos la muestra ocurre como cuando se reducen los intervalos de las sumas de áreas en integración → se tiende al valor de la integral.
- Ejemplo cálculo de volúmenes: Listado 12.7
- Cálculo momentos de inercia: Ejercicio 12.8

Gracias!

Javier Carrasco Serrano

Física Computacional I, Las Tablas