Fundamentos de la programación – Grupo A

Curso 2012–2013

Ejemplo de reorganización en subprogramas

Veamos cómo se reorganizaría el código de una solución del examen del 18 de diciembre utilizando subprogramas. El código inicial es el siguiente:

```
#include <iostream>
using namespace std;
#include <fstream>
#include <iomanip>

int main() {
   ifstream entrada;
   ofstream salida;
   int dato, opcion = -1, buscado, linea;
   bool esPrimo, encontrado;

   system("chcp 1252");
   while (opcion != 0) {
      cout << "1 - Crear la secuencia" << endl;
      cout << "2 - Procesar la secuencia" << endl;
      cout << "3 - Buscar en la secuencia" << endl;
      cout << "0 - Salir" << endl;
      cout << "Opción: ";
      cin >> opcion;
      switch (opcion) {
         case 0:
            break;
         case 1:
            salida.open("datos.txt");
            do {
               cout << "Entero positivo (0 para terminar): ";
               cin >> dato;
               if ((dato >= 1) && (dato <= 1000))
                  salida << dato << endl;
            } while (dato != 0);
            salida << -1;
            salida.close();
            break;
         case 2:
            entrada.open("datos.txt");
            if (!entrada.is_open())
               cout << "¡Error al abrir el archivo!" << endl;
            else {
               entrada >> dato;
               while (dato != -1) {
                  cout << setw(4) << dato << " ";
```

```
                esPrimo = true;
                for (int i = 2; i <= dato / 2; i++) {
                   if (dato % i == 0)
                      if (esPrimo) { // Primer divisor encontrado
                         esPrimo = false;
                         cout << "no es primo (" << i;
                      }
                      else // No es el primer divisor
                         cout << ", " << i;
                }
                if (esPrimo)
                   cout << "es primo";
                else
                   cout << ")";
                cout << endl;
                entrada >> dato;
             }
             entrada.close();
          }
          break;
       case 3:
          entrada.open("datos.txt");
          if (!entrada.is_open())
             cout << "¡Error al abrir el archivo!" << endl;
          else {
             cout << "Cuadrado a buscar: ";
             cin >> buscado;
             encontrado = false;
             linea = 1;
             entrada >> dato;
             while ((dato != -1) && !encontrado)
                if ((dato * dato >= buscado - 10) &&
                    (dato * dato <= buscado + 10))
                   encontrado = true;
                else {
                   entrada >> dato;
                   linea++;
                }
             entrada.close();
             if (encontrado)
                cout << "Encontrado en la línea " << linea << ": " << dato
                     << " (cuadrado: " << dato * dato << ")" << endl;
             else
                cout << "¡Ninguno mayor o igual que el buscado!" << endl;
          }
          break;
       default:
          cout << "Opción incorrecta. Inténtalo de nuevo..." << endl;
    }
  }
  return 0;
}
```

Identifiquemos en el código tareas concretas que se realizan. Buscamos secciones del código que se puedan identificar con una tarea simple, enunciable de forma sencilla.

```
#include <iostream>
using namespace std;
#include <fstream>
#include <iomanip>

int main() {
   ifstream entrada;
   ofstream salida;
   int dato, opcion = -1, buscado, linea;
   bool esPrimo, encontrado;

   system("chcp 1252");
   while (opcion != 0) {
      cout << "1 - Crear la secuencia" << endl;
      cout << "2 - Procesar la secuencia" << endl;
      cout << "3 - Buscar en la secuencia" << endl;
      cout << "0 - Salir" << endl;
      cout << "Opción: ";
      cin >> opcion;
```

Elegir la opción del menú

```
      switch (opcion) {
         case 0:
            break;
         case 1:
            salida.open("datos.txt");
            do {
               cout << "Entero positivo (0 para terminar): ";
               cin >> dato;
               if ((dato >= 1) && (dato <= 1000))
                  salida << dato << endl;
            } while (dato != 0);
            salida << -1;
            salida.close();
            break;
```

Crear la secuencia

```
         case 2:
            entrada.open("datos.txt");
            if (!entrada.is_open())
               cout << "¡Error al abrir el archivo!" << endl;
            else {
               entrada >> dato;
               while (dato != -1) {
                  cout << setw(4) << dato << " ";
                  esPrimo = true;
                  for (int i = 2; i <= dato / 2; i++) {
                     if (dato % i == 0)
                        if (esPrimo) { // Primer divisor encontrado
                           esPrimo = false;
                           cout << "no es primo (" << i;
                        }
                        else // No es el primer divisor
                           cout << ", " << i;
                  }
```

Procesar la secuencia

Procesar el elemento

```
                if (esPrimo)
                   cout << "es primo";
                else
                   cout << ")";
                cout << endl;
                entrada >> dato;
             }
             entrada.close();
          }
          break;
       case 3:
          entrada.open("datos.txt");
          if (!entrada.is_open())
             cout << "¡Error al abrir el archivo!" << endl;
          else {
             cout << "Cuadrado a buscar: ";
             cin >> buscado;
             encontrado = false;
             linea = 1;
             entrada >> dato;
             while ((dato != -1) && !encontrado)
                if ((dato * dato >= buscado - 10) &&
                    (dato * dato <= buscado + 10))
                   encontrado = true;
                else {
                   entrada >> dato;
                   linea++;
                }
             entrada.close();
             if (encontrado)
                cout << "Encontrado en la línea " << linea << ": " << dato
                     << " (cuadrado: " << dato * dato << ")" << endl;
             else
                cout << "¡Ninguno mayor o igual que el buscado!" << endl;
          }
          break;
       default:
          cout << "Opción incorrecta. Inténtalo de nuevo..." << endl;
      }
   }
   return 0;
}
```

Buscar en la secuencia

Podríamos identificar otras subtareas dentro de esas tareas... ¿Hasta dónde debemos llegar?

Las subtareas deben ser perfectamente identificables. Si son extremadamente sencillas y no se repiten, se pueden obviar. Por ejemplo, el código

```
                cout << "Entero positivo (0 para terminar): ";
                cin >> dato;
```

es una tarea bien identificable: *solicitar un entero al usuario*. Pero sólo aparece ahí y quizá no merezca la pena convertir el código en un subprograma. Si se hace tampoco sería incorrecto.

Una vez identificadas las secciones que se trasladarán a subprogramas, debemos identificar las variables que se usan y para cada una descubrir si se trata de un dato local (sólo necesario en esa sección de código) o si se trata de un dato que viene o va a otra parte del programa (parámetros).

Elegir la opción del menú

```
cout << "1 - Crear la secuencia" << endl;
cout << "2 - Procesar la secuencia" << endl;
cout << "3 - Buscar en la secuencia" << endl;
cout << "0 - Salir" << endl;
cout << "Opción: ";
cin >> opcion;
```

Sólo hay un dato, la variable `opcion` (entera). Se utiliza en el `switch` que sigue en el programa principal. Por tanto, se trata de un dato de salida.

Aquí tenemos dos opciones, igualmente válidas: crear una función que devuelva ese valor como resultado o crear un procedimiento con un parámetro por referencia o variable.

Función: será del tipo de dato que se quiere devolver (`int`); requiere una variable local de ese tipo, que al final tendrá el resultado y se devolverá con `return`.

```
int menu() {
   int opcion;
   cout << "1 - Crear la secuencia" << endl;
   cout << "2 - Procesar la secuencia" << endl;
   cout << "3 - Buscar en la secuencia" << endl;
   cout << "0 - Salir" << endl;
   cout << "Opción: ";
   cin >> opcion;
   return opcion;
}
```

La llamada a la función se asignará a la variable `opcion` del programa principal.

Procedimiento: de tipo `void`; tendrá un parámetro por referencia del tipo de dato que se quiere devolver (`int`); se usa el parámetro directamente en el procedimiento; no hay `return`.

```
void menu(int &opcion) {
   cout << "1 - Crear la secuencia" << endl;
   cout << "2 - Procesar la secuencia" << endl;
   cout << "3 - Buscar en la secuencia" << endl;
   cout << "0 - Salir" << endl;
   cout << "Opción: ";
   cin >> opcion;
}
```

En la llamada al procedimiento se pasará la variable `opcion` como argumento.

Crear la secuencia

```
salida.open("datos.txt");
do {
   cout << "Entero positivo (0 para terminar): ";
   cin >> dato;
   if ((dato >= 1) && (dato <= 1000))
      salida << dato << endl;
} while (dato != 0);
salida << -1;
salida.close();
```

Variables: `salida` es un archivo que se abre, se procesa y se cierra en ese código, por lo que será una variable local del subprograma. `dato` se usa para escribir en el archivo y nada más, por lo que será otra variable local. No se requiere ninguna comunicación con el resto del programa, por lo que será un procedimiento sin parámetros:

```
void crearSecuencia() {
   ofstream salida;
   int dato;
   salida.open("datos.txt");
   do {
      cout << "Entero positivo (0 para terminar): ";
      cin >> dato;
      if ((dato >= 1) && (dato <= 1000))
         salida << dato << endl;
   } while (dato != 0);
   salida << -1;
   salida.close();
}
```

Procesar el elemento de la secuencia

```
                cout << setw(4) << dato << " ";
                esPrimo = true;
                for (int i = 2; i <= dato / 2; i++) {
                   if (dato % i == 0)
                      if (esPrimo) { // Primer divisor encontrado
                         esPrimo = false;
                         cout << "no es primo (" << i;
                      }
                      else // No es el primer divisor
                         cout << ", " << i;
                }
                if (esPrimo)
                   cout << "es primo";
                else
                   cout << ")";
                cout << endl;
```

Variables: `dato` (toma valor en el código que procesa la secuencia, por lo que será un dato de entrada; no se modifica, por lo que no es de salida → parámetro por valor); `esPrimo` (sólo se usa en ese fragmento de código → variable local); `i` (local al `for` → se declara directamente en él).

No se devuelve ningún resultado, por lo que será un procedimiento con un parámetro por valor:

```
void procesarDato(int dato) {
   bool esPrimo = true; // Se inicializa en la declaración
   cout << setw(4) << dato << " ";
   for (int i = 2; i <= dato / 2; i++) {
      if (dato % i == 0)
         if (esPrimo) { // Primer divisor encontrado
            esPrimo = false;
            cout << "no es primo (" << i;
         }
         else // No es el primer divisor
            cout << ", " << i;
   }
```

```
   if (esPrimo)
      cout << "es primo";
   else
      cout << ")";
   cout << endl;
}
```

Procesar la secuencia

La tarea anterior es una subtarea de esta. Por tanto, será sustituida en el código de esta por la llamada a su correspondiente subprograma:

```
            entrada.open("datos.txt");
            if (!entrada.is_open())
               cout << "¡Error al abrir el archivo!" << endl;
            else {
               entrada >> dato;
               while (dato != -1) {
                  procesarDato(dato);          Procesar el elemento
                  entrada >> dato;
               }
               entrada.close();
            }
```

Se puede ver que esa sección de código queda ahora mucho más fácil de entender.

Variables: `entrada` es un archivo que se abre, se procesa y se cierra en ese código, por lo que será una variable local del subprograma. `dato` se usa para leer del archivo y nada más, por lo que será otra variable local. No se requiere ninguna comunicación con el resto del programa, por lo que será un procedimiento sin parámetros:

```
void procesarSecuencia() {
   ifstream entrada;
   int dato;
   entrada.open("datos.txt");
   if (!entrada.is_open())
      cout << "¡Error al abrir el archivo!" << endl;
   else {
      entrada >> dato;
      while (dato != -1) {
         procesarDato(dato);
         entrada >> dato;
      }
      entrada.close();
   }
}
```

Buscar en la secuencia

```
            entrada.open("datos.txt");
            if (!entrada.is_open())
               cout << "¡Error al abrir el archivo!" << endl;
            else {
               cout << "Cuadrado a buscar: ";
               cin >> buscado;
               encontrado = false;
               linea = 1;
```

```
            entrada >> dato;
            while ((dato != -1) && !encontrado)
               if ((dato * dato >= buscado - 10) &&
                   (dato * dato <= buscado + 10))
                  encontrado = true;
               else {
                  entrada >> dato;
                  linea++;
               }
            entrada.close();
            if (encontrado)
               cout << "Encontrado en la línea " << linea << ": " << dato
                    << " (cuadrado: " << dato * dato << ")" << endl;
            else
               cout << "¡Ninguno mayor o igual que el buscado!" << endl;
         }
```

Variables: `entrada` es un archivo que se abre, se procesa y se cierra en ese código, por lo que será una variable local del subprograma. `dato` se usa para leer del archivo y nada más, por lo que será otra variable local. `buscado`, `encontrado` y `linea` también parece claro que sólo se usan en esa sección. No se requiere ninguna comunicación con el resto del programa, por lo que será un procedimiento sin parámetros:

```
void busqueda() {
   ifstream entrada;
   int dato, buscado, linea;
   bool encontrado;
   entrada.open("datos.txt");
   if (!entrada.is_open())
      cout << "¡Error al abrir el archivo!" << endl;
   else {
      cout << "Cuadrado a buscar: ";
      cin >> buscado;
      encontrado = false;
      linea = 1;
      entrada >> dato;
      while ((dato != -1) && !encontrado)
         if ((dato * dato >= buscado - 10) &&
             (dato * dato <= buscado + 10))
            encontrado = true;
         else {
            entrada >> dato;
            linea++;
         }
      entrada.close();
      if (encontrado)
         cout << "Encontrado en la línea " << linea << ": " << dato
              << " (cuadrado: " << dato * dato << ")" << endl;
      else
         cout << "¡Ninguno mayor o igual que el buscado!" << endl;
   }
}
```

Sólo queda ver cómo queda el programa principal (la función `main()`) y colocar las funciones en su lugar.

```
#include <iostream>
using namespace std;
#include <fstream>
#include <iomanip>

void crearSecuencia() {
   ofstream salida;
   int dato;
   salida.open("datos.txt");
   do {
      cout << "Entero positivo (0 para terminar): ";
      cin >> dato;
      if ((dato >= 1) && (dato <= 1000))
         salida << dato << endl;
   } while (dato != 0);
   salida << -1;
   salida.close();
}

void procesarDato(int dato) {
   bool esPrimo = true; // Se inicializa en la declaración
   cout << setw(4) << dato << " ";
   for (int i = 2; i <= dato / 2; i++) {
      if (dato % i == 0)
         if (esPrimo) { // Primer divisor encontrado
            esPrimo = false;
            cout << "no es primo (" << i;
         }
         else // No es el primer divisor
            cout << ", " << i;
   }
   if (esPrimo)
      cout << "es primo";
   else
      cout << ")";
   cout << endl;
}

void procesarSecuencia() {
   ifstream entrada;
   int dato;
   entrada.open("datos.txt");
   if (!entrada.is_open())
      cout << "¡Error al abrir el archivo!" << endl;
   else {
      entrada >> dato;
      while (dato != -1) {
         procesarDato(dato);
         entrada >> dato;
      }
      entrada.close();
   }
}
```

```
void busqueda() {
   ifstream entrada;
   int dato, buscado, linea;
   bool encontrado;
   entrada.open("datos.txt");
   if (!entrada.is_open())
      cout << "¡Error al abrir el archivo!" << endl;
   else {
      cout << "Cuadrado a buscar: ";
      cin >> buscado;
      encontrado = false;
      linea = 1;
      entrada >> dato;
      while ((dato != -1) && !encontrado)
         if ((dato * dato >= buscado - 10) &&
             (dato * dato <= buscado + 10))
            encontrado = true;
         else {
            entrada >> dato;
            linea++;
         }
      entrada.close();
      if (encontrado)
         cout << "Encontrado en la línea " << linea << ": " << dato
              << " (cuadrado: " << dato * dato << ")" << endl;
      else
         cout << "¡Ninguno mayor o igual que el buscado!" << endl;
   }
}

int menu() {
   int opcion;
   cout << "1 - Crear la secuencia" << endl;
   cout << "2 - Procesar la secuencia" << endl;
   cout << "3 - Buscar en la secuencia" << endl;
   cout << "0 - Salir" << endl;
   cout << "Opción: ";
   cin >> opcion;
   return opcion;
}

int main() {
   int opcion = -1;
   // Quitamos todas las variables que no se usan en esta función

   system("chcp 1252");

   while (opcion != 0) {
      opcion = menu();
      switch (opcion) {
         case 0:
            break;
```

```
         case 1:
            crearSecuencia();
            break;
         case 2:
            procesarSecuencia();
            break;
         case 3:
            busqueda();
            break;
         default:
            cout << "Opción incorrecta. Inténtalo de nuevo..." << endl;
      }
   }

   return 0;
}
```

En realidad, para que el código anterior sea totalmente correcto, faltaría poner, tras las inclusiones de las bibliotecas, los **prototipos** de los subprogramas. Un prototipo consiste en la cabecera del procedimiento o función terminada en punto y coma (;).

```
#include <iostream>
using namespace std;
#include <fstream>
#include <iomanip>

void crearSecuencia();
void procesarDato(int dato);
void procesarSecuencia();
void busqueda();
int menu();

void crearSecuencia() {
   ofstream salida;
   int dato;
   salida.open("datos.txt");
   ...
```

Se puede ver cómo en este programa ¡¡¡NO HAY NINGUNA VARIABLE GLOBAL!!!