CAPÍTULO 5

Subprogramas: Acciones y funciones

Contenido

5.1 INTRODUCCIÓN

Un subprograma es un pequeño programa dentro de otro programa. Su objetivo es agrupar una serie de sentencias que realizan una tarea concreta bajo un nombre. Ello permite que tareas que puedan ser

```
    leerOperandos(op1, op2);
    cout << "El resultado es: " << op1 + op2 << endl;
}

void restar()
{
    int op1, op2;

    leerOperandos(op1, op2);
    cout << "El resultado es: " << op1 - op2 << endl;
}

void multiplicar()
{
    int op1, op2;

    leerOperandos(op1, op2);
    cout << "El resultado es: " << op1 * op2 << endl;
}

void dividir()
{
    int op1, op2;

    leerOperandos(op1, op2);
    assert(op1 != 0);
    cout << "El resultado es: " << float(op1) / op2 << endl;
}

void leerOperandos(int& op1, int& op2)
{
    cout << "Entra el primer operando: ";
    cin >> op1;
    cout << "Entra el segundo operando: ";
    cin >> op2;
}
```

El resultado de una ejecución podría ser el siguiente:

```
1.- Sumar
2.- Restar
3.- Multiplicar
4.- Dividir
0.- Salir
Elige una opcion (0-4): 1
Entra el primer operando: 30
Entra el segundo operando: 40
El resultado es: 70
1.- Sumar
2.- Restar
3.- Multiplicar
```

```
// Función que determina si dos vértices de reales son iguales
// Nótese que se utiliza un esquema ligeramente diferente porque
// dos reales supuestamente iguales se pueden almacenar con algún
// decimal diferente en función de la precisión, aquí asumimos que
// dos reales con una diferencia menor de 0.000001 son iguales
bool heAcabado(double x0, double y0, double xFin, double yFin)
{
     const double precision = 0.000001;

     return (fabs(x0 - xFin) < precision
          && fabs(y0 - yFin) < precision);
}

// Calcula la caja englobante (genérico)
template <class T>
void cajaEnglobante(T& coordX0, T& coordY0, T& coordXFin,
                    T& coordYFin)
{
     T x, y, xIni, yIni;
     leerVertice(xIni, yIni);
     coordX0 = coordXFin = xIni;
     coordY0 = coordYFin = yIni;

     leerVertice(x, y);

     while (!heAcabado(xIni, yIni, x, y))
     {
          coordX0 = minimo(x, coordX0);
          coordY0 = minimo(y, coordY0);
          coordXFin = maximo(x, coordXFin);
          coordYFin = maximo(y, coordYFin);

          leerVertice(x, y);
     }
}

// Calcula el mínimo de dos elementos
template <class T>
T minimo(T x, T y)
{
     T aux = x;

     if (y < aux)
          aux = y;

     return aux;
}

// Calcula el máximo de dos elementos
```