varios niveles intermedios. En los programas se debe ver cada mensaje y tomar una determinación. Siempre es mejor tratar de escribir todos los programas sin que aparezca ningún mensaje de advertencia o de error. (Con un mensaje de error el compilador no creará el archivo ejecutable.)

Taller

El taller le proporciona preguntas que le ayudarán a afianzar su comprensión del material tratado así como ejercicios que le darán experiencia en el uso de lo aprendido. Trate de comprender el cuestionario y dé las respuestas antes de continuar al siguiente capítulo. Las respuestas se proporcionan en el apéndice G, "Respuestas".

Cuestionario

1. Dé tres razones por las cuales el C es la mejor selección de lenguaje de programación.
2. ¿Qué hace el compilador?
3. ¿Cuáles son los pasos en el ciclo de desarrollo en el programa?
4. ¿Qué comando se necesita teclear para compilar un programa llamado PROGRAM1.C en su compilador?
5. ¿Su compilador ejecuta el enlazado y la compilación con un solo comando o se tienen que dar comandos separados?
6. ¿Qué extensión se debe usar para los archivos fuente del C?
7. ¿Es FILENAME.TXT un nombre válido para un archivo fuente del C?
8. Si se ejecuta un programa que se ha compilado y no funciona como se esperaba, ¿qué se debe hacer?
9. ¿Qué es el lenguaje de máquina?
10. ¿Qué hace el enlazador?

Ejercicios

1. Use el editor de texto para ver el archivo objeto creado por el listado 1.1. ¿Se parece el archivo objeto al archivo fuente? (No guarde este archivo cuando salga del editor.)
2. Teclee el siguiente programa y compílelo. ¿Qué hace este programa? (No incluya los números de línea.)

```
1: #include <stdio.h>
2:
3: int radius, area;
4:
5: main()
6: {
7:     printf( "Enter radius (i.e. 10): " );
8:     scanf( "%d", &radius );
9:     area = 3.14159 * radius * radius;
10:    printf( "\n\nArea = %d", area );
11:    return 0;
12: }
```

3. Teclee y compile el siguiente programa. ¿Qué hace este programa?

```
1: #include <stdio.h>
2:
3: int x,y;
4:
5: main()
6: {
7:     for ( x = 0; x < 10; x++, printf( "\n" ) )
8:         for ( y = 0; y < 10; y++ )
9:             printf( "X" );
10:
11:    return 0;
12: }
```

4. BUSQUEDA DE ERRORES: El siguiente programa tiene un problema. Tecléelo en el editor y compílelo. ¿Qué línea genera mensajes de error?

```
1: #include <stdio.h>
2:
3: main();
4: {
5:     printf( "Keep looking!" );
6:     printf( "You\'ll find it!" );
7:     return 0;
8: }
```

5. BUSQUEDA DE ERRORES: El siguiente programa tiene un problema. Tecléelo en el editor y compílelo. ¿Qué línea da problemas?

```
1: #include <stdio.h>
2:
3: main()
4: {
5:     printf( "This is a program with a " );
6:     do_it( "problem!");
7:     return 0;
8: }
```

6. Haga los siguientes cambios al programa del ejercicio número 3. Vuélvalo a compilar y ejecute este programa. ¿Qué hace ahora el programa?

```
9:              printf( "%c", 1 );
```

7. Teclee y compile el siguiente programa. Este programa puede usarse para imprimir sus listados. Si se tienen errores, asegúrese de haber tecleado el programa correctamente.

   El uso de este programa es PRINT_IT nombre de *archivo.ext*, donde nombre de archivo.ext es el nombre de archivo fuente junto con su extensión. Observe que este programa añade números de línea al listado. (No se preocupe por la longitud de este programa; no espero que lo entienda todavía. Se incluye aquí para ayudarle a comparar las impresiones de sus programas con las que se dan en el libro.)

```
1:  /* PRINT_IT.C- Este programa imprime un listado con números de
           línea*/
2:
3:  #include <stdio.h>
4:
5:  void do_heading(char *filename);
6:
7:  int line, page;
8:
9:  main( int argv, char *argc[] )
10: {
11:   char buffer[256];
12:   FILE *fp;
13:
14:   if( argv < 2 )
15:   {
```

```
16:     fprintf(stderr, "\nProper Usage is: " );
17:     fprintf(stderr, "\n\nPRINT_IT filename.ext\n" );
18:       exit(1);
19:    }
20:
21:    if (( fp = fopen( argc[1], "r" )) == NULL )
22:    {
23:      fprintf( stderr, "Error opening file, %s!", argc[1]);
24:        exit(1);
25:    }
26:
27:    page = 0;
28:    line = 1;
29:    do_heading( argc[1]);
30:
31:    while( fgets( buffer, 256, fp ) != NULL )
32:    {
33:       if( line % 55 == 0 )
34:          do_heading( argc[1] );
35:
36:       fprintf( stdprn, "%4d:\t%s", line++, buffer );
37:    }
38:
39:    fprintf( stdprn, "\f" );
40:    fclose(fp);
41:    return 0;
42:    }
43:
44:    void do_heading( char *filename )
45:    {
46:     page++;
47:
48:     if ( page > 1)
49:        fprintf( stdprn, "\f" );
50:
51:     fprintf( stdprn, "Page: %d, %s\n\n", page, filename );
52:    }
```