Introducción al C++

Héctor Ferrada
académico

Instituto de Informática
Universidad Austral de Chile
INFO088 - Estructuras de Datos y Algoritmos

2Do Semestre, 2019

El Lenguaje de Programación C++

Se puede considerar como un superconjunto del C, que nació para añadirle cualidades y características de las que carecía.
C++ es un lenguage compilado y de propósito general, también es de *libre forma* al ser: procedural, orientado a objetos y de programación genérica. Además:

- C++ es un lenguaje imperativo orientado a objetos derivado del C, aunque no es un lenguaje orientado a objetos puro (como lo es Java por ejemplo). C++ es clasificado como de nivel intermedio, combinado características de lenguages de alto y de bajo nivel.
- Reutiliza la mayoría de las funcionalidades del C; y generalmente puede compilarse un programa C bajo C++, pero no a la inversa si este utiliza características especiales del C++.
- La compilación con GNU gcc/g++ dispone de varias opciones de optimización (p.ej. -O3)

C++, Otras Características

- Sensible a mayúsculas.
- Tiene un conjunto completo de instrucciones de control.
- Permite la agrupación de instrucciones.
- La E/S no forma parte del lenguaje, sino que se proporciona a través de una biblioteca de funciones.
- Permite la separación de un programa en módulos que admiten compilación independiente y hace un uso extensivo de llamadas a funciones.
- Uso de librerías estándares incluidas en el sistema (en el espacio de nombre *std*), también de la STL con el uso de plantillas.
- Permite la programación de bajo nivel (nivel bit).
- Implementación de apuntadores: el uso extensivo de apuntadores para la memoria, arreglos, estructuras y funciones.
- Es de tipado fuerte y estático: las variables deben asociarse a un tipo defino de datos y las comprobaciones de tipo se hacen en tiempo de compilación.

Razones de la Gran Popularidad del C++

- Uso de constructores de alto nivel.
- Programación de código de bajo-nivel.
- Tipicamente, se crean programas más eficientes que al usar otros lenguajes de programación.
- La posibilidad de poder ser compilado en una variedad de computadoras, con pocos cambios (portabilidad).

Un punto en contra es que tiene una detección pobre de errores, lo cual en ocasiones es problemático para los principiantes.

Nuestro primer programa en C++

Con cualquier editor cree un archivo prog1.cpp y escriba:

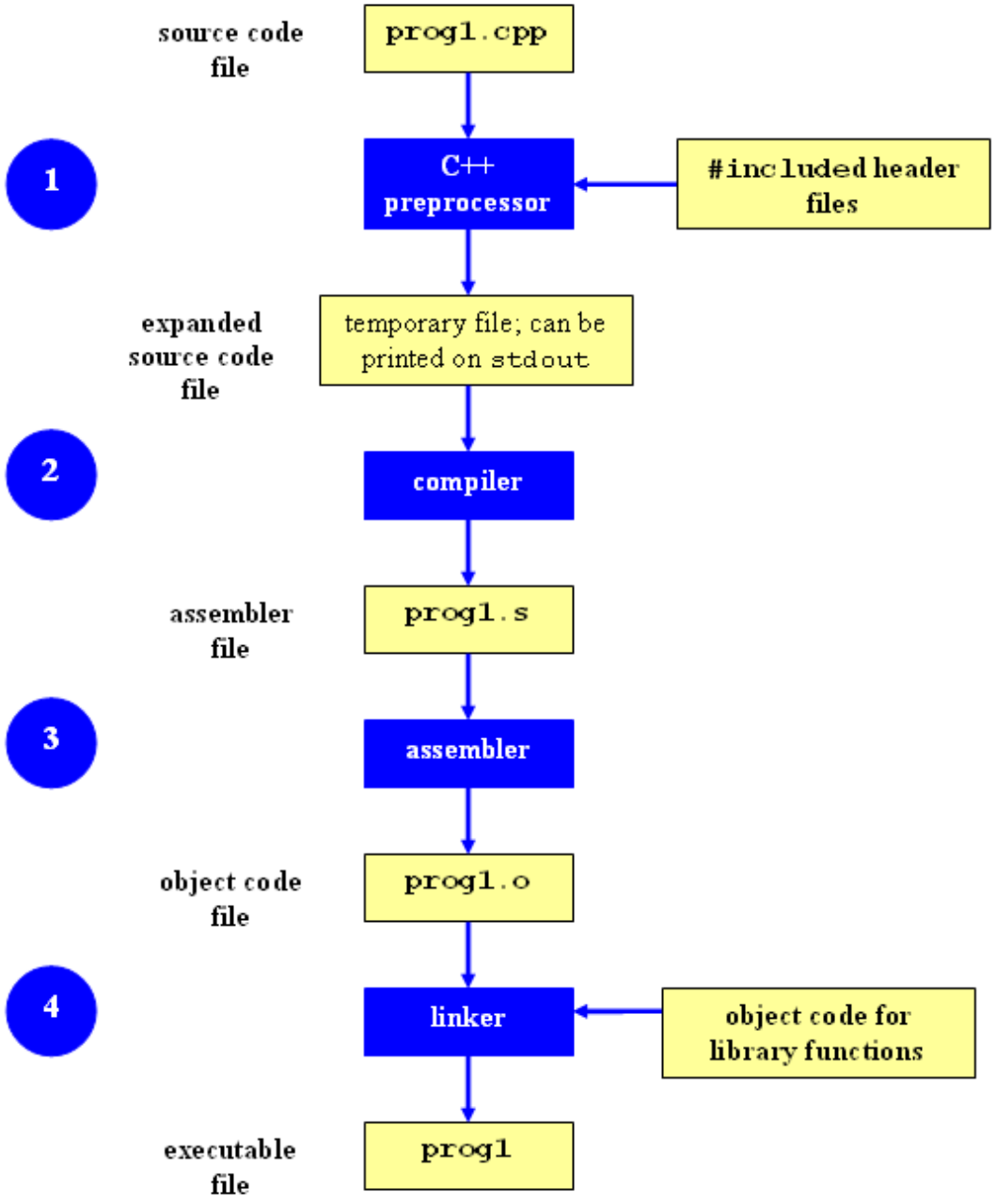
```
#include <iostream>   //Biblioteca donde se encuentra la función cout
using namespace std;  //uso del espacio de nombre std

int main()
{
   int x = 2;           // un entero
   float y = 3.1415;    // un real
   char c = 'a';        // un caracter
   char curso[10] = "INFO053"; // una cadena, en realidad es un arreglo de char's
   string s;            // un string, posee soporte en <string> (parte de la librería
stanfdard de C++)
                        // para varias funciones típicas con cadenas de caracteres

   cout << "Hola! Estamos comenzando el curso " << curso << endl;               //
Imprime
   cout << "Ahora leemos algo desde la entrada estándar (teclado)." << endl;
   cout << "Ingrese un Número entero: ";
   cin >> x;                                                                    // lee

   cout << "Ingrese una frase: ";
   cin >> s;

   cout << "Has ingresado x = " << x << " y s = " << s;

   return 0;
}
```

El Proceso de Compilación en C++

La etapa de compilación consiste en tomar un código fuente, p.ej. prog1.cpp, y aplicar una serie de transformaciones hasta obtener finalmente el archivo ejecutable de dicho fuente:

g++ -Wall -std=c++14 -o prog1 prog1.cpp

Donde:
g++ (obligatorio) es el comando de compilación en c++ (este compilador pertenece a la GNU Compiler Collection).
-Wall (opcional) arrojará por pantalla todos los mensajes de advertancia.
-std (opcional) indica que se esta usando la librería estándar del lenguaje c++14 (usaremos esta std durante todo el curso).
-o prog1 (opcional) es el nombre del ejecutable final creado a partir de prog1.cpp. Si este flag no esta presente, la salida será **a.out**.


source code file
prog1.cpp
1
C++ preprocessor
#included header files
expanded source code file
temporary file; can be printed on stdout
2
compiler
assembler file
prog1.s
3
assembler
object code file
prog1.o
4
linker
object code for library functions
executable file
prog1

... Compilación en C++

Si bien todo este proceso puede invocarse con una sola línea de comando (como en la anterior slide), la secuencia completa consta de 4 etapas, las que podemos hacer por separado:

- **I.- Preprocesamiento.** (tarea del *preprocessor*).

Aquí, al usar el flag -**E**, se reemplazan las directivas de preprocesamiento por código puro en C++. Dichas directivas son las que se encuentran declaradas inmediatamente después de símbolos # en el fuente original, tales como: #define, #if, #include, etc.
Como salida de esta tapa, tendremos un archivo prog1.cpp más grande que el anterior y escrito completamente en código C++ puro. Ejemplo:

g++ -Wall -std=c++14 -E prog1.cpp (prog1.cpp se expande)
g++ -Wall -std=c++14 -E -o temp.cpp prog1.cpp

Ejecute la segunda opción y luego con un editor revise que hay en el archivo temp.cpp

... Compilación en C++

- **II.- Compilación.** (tarea del *compiler*).

Al usar el flag -**S**, El archivo expandido resultante de la etapa anterior es traducido a lenguaje ensamblador (lenguaje de bajo nivel), creando un archivo con la extensión .s con el mismo nombre del fuente.
Comando para obtener el .s a partir del fuente:

g++ -Wall -std=c++14 -S prog1.cpp
g++ -Wall -std=c++14 -S -o prog1.s temp.cpp

Ejecute la segunda opción y luego con un editor revise que hay en el archivo prog1.s

... Compilación en C++

- **III.- Ensamblado.** (tarea del *assembler*).

Al usar el flag -**c**, el archivo en código ensamblador generado por el compilador, es traducido a código objeto (lenguaje de máquina). Comando:

g++ -Wall -std=c++14 -c prog1.cpp
g++ -Wall -std=c++14 -c -o prog1.o prog1.s

Ejecute y revise que hay en el archivo prog1.o (ambas opciones crean el archivo objeto prog1.o)

... Compilación en C++

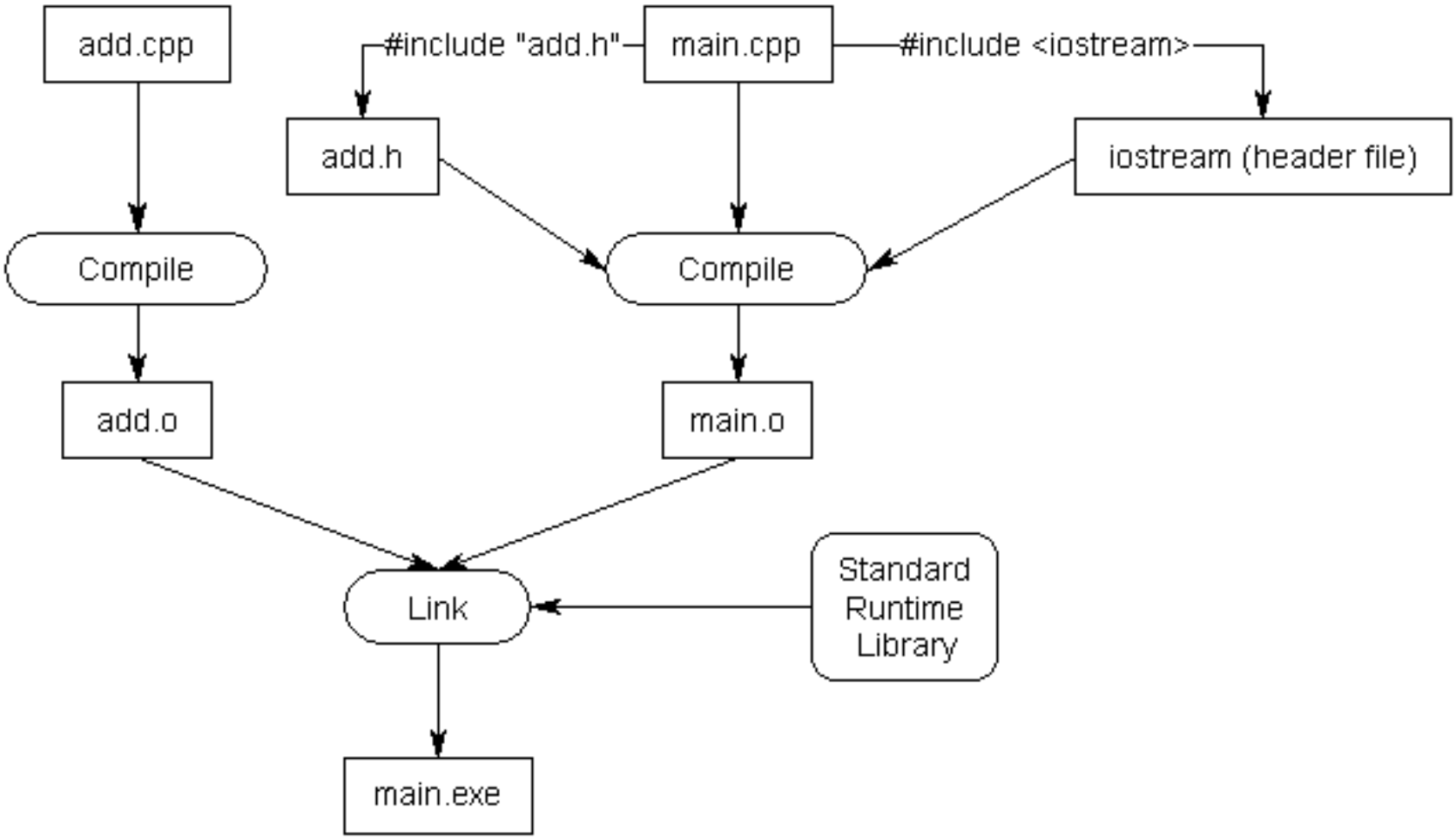
- **IV.- Enlazado.** (tarea del *linker*).

Pueden existir funciones de C/C++ incluidas en nuestro código, tal como printf() (heredada del C original), que ya se encuentran compiladas y ensambladas en bibliotecas externas a su programa pero existentes en el sistema. Es preciso incorporar el código binario (ejecutable) de estas funciones a nuestro ejecutable. En esto consiste la etapa de enlace, donde se reúnen uno o más módulos en código objeto con el código existente en las bibliotecas tanto del sistema como otras posibles librerías creadas por el usuario, creando un solo archivo ejecutable.
Comando:

g++ -Wall -std=c++14 -o prog1 prog1.o

Vea que hay en el ejecutable prog1. Luego, si lo requiere, de permisos de ejecución a prog1 (use chmod +x prog1) y ejecute:
> ./prog1

Compilile and Link in C++



Segundo programa en C++

Este código, prog2.cpp, realiza lo siguiente:
i. Valida que se reciban dos argumentos como entrada al programa: el 1ro es el nombre del programa y el 2do es un string.
Importante: todo lo que se recibe como argumento son strings.
ii. A modo de ejemplo, se declaran unas variables de diferentes tipos de datos. Si incluimos el flag -Wall, el compilador arrojará unos **warnings**, y no dirá que no estamos usamos todas las variables en el resto del programa.
iii. Leemos desde la consola un string, con capacidad de 100 caracteres, y lo guardamos en un arreglo de caracteres.
iv. Leemos un intervalo de enteros $[a, b]$, validando que $a \leq b$.
v. Invocamos a la función suma_pares_impares() que realiza el cálculo de la suma de los enteros pares en el intervalo y la suma de los enteros impares en el intervalo.
Luego compíle con: **g++ -Wall -std=c++14 -o prog2 prog2.cpp**
Ejecute con: **./prog2**

```cpp
#include <iostream>     // Biblioteca donde se encuentra la función cout
#include <string.h>     // para uso del strcpy()
using namespace std;    // uso del espacio de nombre std

// Declaración de función
void suma_pares_impares(int a,  int b, int* par, int *impar);

// A esta definición del main(), se le pueden pasar argumentos
int main(int argc, char **argv){
        int a, b, sum_par, sum_impar;

        if(argc != 2){
                cout << "Error. Debe ejecutarse como ./prog2 string" << endl;
                exit(EXIT_FAILURE);
        }

        // Ejemplo de otros tipos de datos
        float c = 3.2f;
        double d = 3.3;
        char q = 'a';

        // crear un string (que es un arreglo en realidad)
        char *s = new char[100];
        strcpy(s, argv[1]);
        cout << "Estring leído desde los argumentos de ejecución: " << s << endl;

        a=1; b=0;
        while(a>b){
                cout << "Ingrese el inicio del intervalo, a :";
                cin >> a;
                cout << "Ingrese el fin del intervalo, b :";
                cin >> b;
        }

        suma_pares_impares(a, b, &sum_par, &sum_impar);
        cout << "Suma de pares en [" <<a<< ".." <<b<< "] = " << sum_par << endl;
        cout << "Suma de impares en [" <<a<< ".." <<b<< "] = " << sum_impar << endl;

        return EXIT_SUCCESS;
}
```

Procedimiento suma_pares_impares

```
// suma pares e impares en el intervalo [a..b], en *par y *impar respectivamente
void suma_pares_impares(int a, int b, int* par, int* impar){
        int sum_aux = 0;
        *par = 0;

        while(a<=b){
                if(a%2==0)
                        (*par)+=a;      // sumar directamente usando la variable par
                else                    // (es más lento, hay un doble acceso a
memoria)
                        sum_aux+=a;     // la otra forma usa una variable temporal
                a++;
        }
        *impar = sum_aux;               // actualizamos el punteros antes de retornar
}
```

Estructuras de Control en C++

- Sentencia *if - else*
- Sentencia de selección *switch*
- Ciclo de iteración *while*
- Ciclo de iteración *for*

Ver documento estructurasDeControl.pdf

Trabajando en Clases

Codifique en C++ un programa para cada uno de los siguientes problemas:

1. Lea desde el teclado 3 números enteros e imprimalos en orden de menos a mayor.
2. Lea desde el teclado un número entero positivo e imprima por pantalla sus dígitos de derecha a izquierda. Realice esto dentro de una función. Repita la acción pero ahora imprimiendo los dígitos de izquierda a derecha .
3. Reciba, desde los argumentos del programa, dos número enteros, l y r, y cree un código que analice cada entero en $[l, r]$ e imprima aquellos que son primos.
4. Escriba un programa que pida el ingreso de un número entero y lo imprima en orden reverso. Trabaje solo con datos tipo int.
5. Escriba un programa que pida el ingreso de un número entero (en base 10), lo transforme a binario y lo imprima.
6. Escriba un programa que pida el ingreso de un número entero y chequee si el número el palíndrome. Un número es palíndrome si al leer sus dígitos de derecha a izquierda se obtiene el mismo número. Ejemplos: 22 ,33, 44, 121, 12321, 131, etc. Trabaje solo con datos tipo int.