



UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO



FACULTAD
DE INGENIERÍA

Programación Orientada a Objetos

Cambios de C++ respecto de C

Mg. Ing. César Aranda

cesar.aranda@ingenieria.uncuyo.edu.ar
unidatos@gmail.com

Ingeniería en Mecatrónica

Objetivos

Introducir el lenguaje C++ a partir de los conocimientos adquiridos del lenguaje C

Actualizar o ampliar el conocimiento de las sentencias básicas mediante el análisis de porciones de código

Resolver problemas sencillos usando sentencias básicas

Contenidos

- El lenguaje C++ como extensión de C. Generalidades.
- Tipos de datos. Variables. Constantes
- Operadores y Expresiones
- Casting
- Punteros y Referencias
- Declaración de variables
- Flujos de entrada/salida (estándar)
- Estructuras de Control

Ing. César Aranda

3

Tipos de Datos

Nombre	Descripción	Tamaño*	Rango de valores*
char	Carácter o entero pequeño.	1byte	con signo: -128 to 127 sin signo: 0 a 255
short int (short)	Entero corto.	2bytes	con signo: -32768 a 32767 sin signo: 0 a 65535
int	Entero.	4bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
long int (long)	Entero largo.	4bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
bool	Valor booleano. Puede tomar dos valores: verdadero o falso.	1byte	true o false
float	Número de punto flotante.	4bytes	3.4e +/- 38 (7 dígitos)
double	De punto flotante de doble precisión.	8bytes	1.7e +/- 308 (15 dígitos)
long double	Long de punto flotante de doble precisión.	8bytes	1.7e +/- 308 (15 dígitos)

Los valores dependen de la arquitectura utilizada
 Se muestran valores típicos en una computadora de 32 bits

Ing. César Aranda

4

De C a C++: Archivos de Cabecera

Las bibliotecas de C++:

Tienen archivos de cabecera sin extensión **.h**

cctype en lugar de ~~cctype.h~~

cmath en lugar de ~~math.h~~

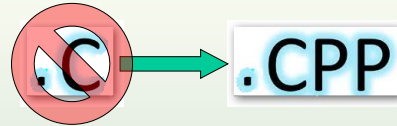
cstdlib en lugar de ~~stdlib.h~~

...

Admiten usar **namespace**. Ésta es una declaración de región, es una sección con nombre dentro de las que se incluyen las declaraciones. El propósito es definir un identificador de localización para evitar colisiones.

Ejemplo de uso:

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



Ing. César Aranda

5

De C a C++: Declaración simplificada

Sean los tipos de datos:

```
enum color {ROJO, AMARILLO, VERDE};
```

```
struct material{
    long cod_mat;
    char nombre[20];
}
```

En C hay que declarar:

```
enum color frente, fondo;
struct material soporte, palanca, resorte;
```

En C++ solamente:

```
color frente, fondo;
material soporte, palanca, resorte;
```

Ing. César Aranda

6

Lectura y escritura estándar

El archivo de cabecera `iostream` de la biblioteca de C++ proporciona:

Un flujo de entrada estándar `cin` y un operador de extracción `>>`, para tomar valores del flujo y almacenarlos en variables.

Sintaxis de la sentencia:

`cin >>` Lista de variables;

Familias de C
`scanf ()`
`getch ()`
`kbhit ()`

Un flujo de salida estándar `cout` y un operador de inserción `<<`, para recuperar valores de las variables y colocarlos en el flujo.

Sintaxis de la sentencia:

`cout <<` Lista de variables;

Familias de C
`printf ()`
`putchar ()`

Ing. César Aranda

7

Ejemplo con E/S

```
2  * File:  ejem-io-02.cpp
3
4  * Programa que solicita la base y la altura de un triángulo,
5  * calcula su área y muestra los datos y el resultado. */
6
7  #include <iostream>
8  using namespace std;
9
10 int main() {
11     float base, altura, area;
12
13     cout << "Base (cm)? : ";
14     cin >> base;
15     cout << "Altura (cm) ? : ";
16     cin >> altura;
17
18     area = base * altura / 2;
19
20     cout << "El area de un triángulo de base " << base
21          << "cm y altura " << altura << "cm es de " << area << "cm2";
22
23     return EXIT_SUCCESS;
24 }
```

Ing. César Aranda

8

Jerarquía de Operadores (1)

Operador	Descripción
<code>...</code>	Resolución de ámbito
<code>--</code>	Post- incremento y decremento
<code>()</code>	Llamada a función
<code>[]</code>	Elemento de vector
<code>.</code>	Selección de elemento por referencia
<code>-></code>	Selección de elemento con puntero
<code>++ --</code>	Pre- incremento y decremento
<code>+ -</code>	Suma y resta unitaria
<code>! ~</code>	NOT lógico y NOT binario
<code>(type)</code>	Conversión de tipo
<code>*</code>	Indirección
<code>&</code>	Dirección de
<code>sizeof</code>	Tamaño de
<code>new[]</code>	Asignación dinámica de memoria
<code>delete[]</code>	Desasignación dinámica de memoria

Ing. César Aranda

9

Jerarquía de Operadores (2)

Operador	Descripción
<code>.* ->*</code>	Puntero a miembro
<code>* / %</code>	Multiplicación, división y módulo
<code>+ -</code>	Suma y resta
<code><< >></code>	Operaciones binarias de desplazamiento
<code>< <=</code> <code>> >=</code>	Operadores relaciones "menor que", "menor o igual que", "mayor que" y "mayor o igual que"
<code>== !=</code>	Operadores relaciones "igual a" y "distinto de"
<code>&</code>	AND binario
<code>^</code>	XOR binario
<code> </code>	OR binario
<code>&&</code>	AND lógico
<code> </code>	OR lógico

Ing. César Aranda

10

Jerarquía de Operadores (3)

Operador	Descripción
$c? t: f$	Operador ternario
$=$ $+= -=$ $*= /= \%=$ $<<= >>=$ $\&= ^= =$	Asignaciones
<code>throw</code>	lanzamiento de excepciones
<code>,</code>	Coma

try {
}
catch {}

Ing. César Aranda

11

Ejemplo con operador ::

```
6 #include <iostream>
7 using namespace std;
8
9 int main(int argc, char** argv) {
10     cout << "Un ejemplo muy simple sin Hola Mundo !!?" << endl;
11     return 0;
12 }
```

```
6 #include <iostream>
7
8 int main(int argc, char** argv) {
9     std::cout << "Un ejemplo muy simple sin Hola Mundo !!?"
10     << std::endl;
11     return 0;
12 }
```

```
ejem-spacename-01.cpp: In function 'int main(int, char**)':
"/usr/bin/make" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .build-conf
ejem-spacename-01.cpp:9: error: 'cout' undeclared (first use this function)
make[1]: Entering directory '/cygdrive/c/Documents and Settings/Cesar/Mis documentos/NetBear'
"/usr/bin/make" -f nbproject/Makefile-Debug.mk dist/Debug/Cygwin-Windows/verificacion.exe
make[2]: Entering directory '/cygdrive/c/Documents and Settings/Cesar/Mis documentos/NetBear'
ejem-spacename-01.cpp:9: error: (Each undeclared identifier is reported only once for each
ejem-spacename-01.cpp:9: error: 'endl' undeclared (first use this function)
make[2]: *** [build/Debug/Cygwin-Windows/ejem-spacename-01.o] Error 1
make[1]: *** [.build-conf] Error 2
make: *** [.build-impl] Error 2
```

Ing. César Aranda

12

Ejemplo: Conversiones

```
1 #include <iostream>
2 using namespace std;
3
4 int main(int argc, char *argv[]) {
5
6     int x = 10, y = 3;
7
8     cout << "Resultado entero de " << x << "/" << y << ": " << x / y << endl;
9
10
11     cout << "Resultado real de " << x << "/" << y << ": " << (float)x / y << endl;
12
13
14     cout << "Resultado real de " << x << "/" << y << ": " << float(x) / y << endl;
15
16     return 0;
17 }
```

```
CA\Program Files (x86)\Zinja\bin\runner.exe
Resultado entero de 10/3: 3
Resultado real de 10/3: 3.33333
Resultado real de 10/3: 3.33333
```

Ing. César Aranda

13

Ejemplo: Valor por Defecto de Función

```
6 #include <iostream>
7 #include <cmath>
8 using namespace std;
9
10 double VelocidadFinal(double h, double v0 = 0.0, double g = 9.8);
11
12 int main() {
13     cout << "Velocidad final en caída libre desde 100 metros,\n" <<
14         "partiendo del reposo en la Tierra: " <<
15         VelocidadFinal(100) << "m/s" << endl << endl;
16     cout << "Velocidad final en caída libre desde 100 metros,\n" <<
17         "con una velocidad inicial de 10m/s en la Tierra: " <<
18         VelocidadFinal(100, 10) << "m/s" << endl << endl;
19     cout << "Velocidad final en caída libre desde 100 metros,\n" <<
20         "partiendo del reposo en la Luna: " <<
21         VelocidadFinal(100, 0, 1.6) << "m/s" << endl << endl;
22     cout << "Velocidad final en caída libre desde 100 metros,\n" <<
23         "con una velocidad inicial de 10m/s en la Luna: " <<
24         VelocidadFinal(100, 10, 1.6) << "m/s" << endl;
25     return 0;
26 }
27
28 double VelocidadFinal(double h, double v0, double g) {
29     double t = (-v0 + sqrt(v0 * v0 + 2 * g * h)) / g;
30     return v0 + g*t;
31 }
```

Ing.

14

Ejemplo: Referencias con puntero

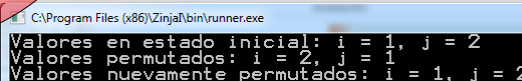
Programa con función para permutar 2 valores por referencia

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

void permutar(int *a, int *b);

int main() {
    int i = 1, j = 2;
    cout << "Valores en estado inicial: i = " << i << ", j = " << j << endl;
    permutar(&i, &j);
    cout << "Valores permutados: i = " << i << ", j = " << j << endl;
    permutar(&i, &j);
    cout << "Valores nuevamente permutados: i = " << i
        << ", j = " << j << endl;
    return 0;
}

void permutar(int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```



Ing. César Aranda

15

Ejemplo: Función y Referencias en C++

Programa con función para permutar 2 valores por referencia

```
6  #include <iostream>
7  using namespace std;
8  void permutar(int &a, int &b);
9
10
11 int main() {
12     int i = 1, j = 2;
13     cout << "Valores en estado inicial: i = " << i << ", j = " << j << endl;
14     permutar(i, j);
15     cout << "Valores permutados: i = " << i << ", j = " << j << endl;
16     permutar(i, j);
17     cout << "Valores nuevamente permutados: i = " << i
18         << ", j = " << j << endl;
19     return 0;
20 }
21
22 void permutar(int &a, int &b) {
23     int temp;
24     temp = a;
25     a = b;
26     b = temp;
27 }
```

Evita el uso de punteros y de los operadores de dirección e indirección

Ing. César Aranda

16

Variables de tipo Referencia

Son variables normales

Las variables referencia se declaran por medio del tipo de dato seguido del carácter **&**

Deben ser inicializadas a otra variable o a un valor numérico o alfanumérico

Una vez que una variable referencia ha sido declarada como alias de una variable no puede ser declarada como alias de otra variable

No confundir el uso de & en la declaración de una referencia con el operador dirección &

Ejemplo:

```
int i=2;  
int& iref = i; // declaración de referencia válida  
int& jref;      // declaración de referencia no válida
```

Ing. César Aranda

17

Ejemplo: Reserva Dinámica

Programa que reserva memoria dinamica para un vector (array) de caracteres

```
6  #include <iostream>  
7  #include <string>  
8  using namespace std;  
9  
10 int main() {  
11     char Nombre[50];  
12     cout << "Introduzca un Nombre:";  
13     cin >> Nombre;  
14  
15     char *Copia = new char[strlen(Nombre) + 1];  
16     strcpy(Copia, Nombre);  
17     cout << Copia;  
18  
19     delete [] Copia;  
20     return EXIT_SUCCESS;  
21 }
```

Ing. César Aranda

18

estát. ← cosa.nombre
dinám. ← cosa → nombre

Gestión Dinámica de Memoria

Uso de operadores `new` y `delete`

La variable creada con `new` perdura hasta que es explícitamente borrada con el operador `delete`

Puede traspasar la frontera de su bloque y ser manipulada por instrucciones de otros bloques

Permite crear variables de cualquier tipo

`new` devuelve un puntero a la variable creada

Ejemplo:

```
struct material{
    long cod_mat;
    char nombre[20];
}
material* un_tablero;
un_tablero = new material;
```

Ing. César Aranda

19

Ejemplo: para Estructuras de Control

```
6  #include <iostream>
7  using namespace std;
8
9  enum direccion {
10     NINGUNA, ADELANTE, DERECHA, ATRAS, IZQUIERDA, ARRIBA, ABAJO
11 };
12
13 void cambiar_nivel(int hacia);
14 void moverse(int hacia);
15 void doblar(int hacia);
16 void invertir();
17 void avanzar();
18
19 int main() {
20     int op_mover;
21     cout << "Elija la dirección de movimiento"
22         << "[ 0 (cero) para TERMINAR ]" << endl
23         << "1. ADELANTE" << endl
24         << "2. DERECHA" << endl
25         << "3. ATRAS" << endl
26         << "4. IZQUIERDA" << endl
27         << "5. ARRIBA" << endl
28         << "6. ABAJO" << endl
29         << "0. SALIR" << endl;
30     cin >> op_mover;
```

Ing. Césa

20

Ejemplo: Decisiones

```
32     if (op_mover == NINGUNA) {  
33         cout << "Juego terminado" << endl;  
34         return 0;  
35     }  
  
37     if (op_mover == ARRIBA || op_mover == ABAJO) {  
38         cambiar_nivel(op_mover);  
39     } else {  
40         moverse(op_mover);  
41     }  
42  
43     return 0;  
44 }  
  
46 void moverse(int hacia) {  
47     switch (hacia) {  
48         case DERECHA: doblar(DERECHA);  
49         break;  
50         case ATRAS: invertir();  
51         break;  
52         case IZQUIERDA: doblar(IZQUIERDA);  
53         break;  
54         default: avanzar();  
55     }  
56 }
```

Ing. César Aranda

21

Referencias

- STROUSTRUP, B. (2009): [Programming Principles & Practices using C++](#), Addison Wesley
- DEITEL, H. M. y DEITEL, P. J. (2009): [Cómo Programar en C/C++](#), 6ª edición, Prentice-Hall
- ACERA GARCIA, M.A. (2010): [C/C++](#), Anaya Multimedia
- ELLIS, M.A. y STROUSTRUP, B. (2004): [C++ Manual de Referencia](#), Ed. Diaz De Santos
- Lecturas Complementaria en archivos:
Lectura_01.pdf, Lectura_03.pdf
- URLs:
<http://c.conclase.net/>
<http://www.cprogramming.com/>
<http://www.cplusplus.com/doc/tutorial/>
<http://www.desy.de/gna/html/cc/Tutorial/tutorial.html>
http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C%2B%2B
http://www.zator.com/Cpp/E_Ce.htm
<http://www.programacionenc.net/>
http://foro.elhacker.net/programacion_cc-b49.0/
http://www.lawebdelprogramador.com/cursos/C_Visual_C/index1.html

Ing. César Aranda

22