Estándar de Codificación para C++
Departamento de Ciencias Computacionales
Escuela de Ingeniería y Tecnologías de Información
Campus Monterrey
Comité de Programación
Tecnológico de Monterrey Campus Monterrey

Antecedentes

Entre los principales propósitos de tener una convención de código están: tener una visión consistente del código de manera que el lector pueda enfocarse en el contenido; entender el código más rápidamente; facilitar el mantenimiento y aplicar las mejores prácticas del desarrollo.

Lineamientos

Los lineamientos de codificación son:

Tema	Lineamiento
Nombres de archivo que almacena el programa	• Utilizar la notación PascalCase: La primer letra en cada palabra debe ser mayúscula (NombreDescriptivo). • Debe ser un nombre descriptivo del propósito del programa. • No debe incluir caracteres especiales ni acentos. • Si requiere el uso de dos o más palabras cada una debe iniciar con mayúscula. • Se debe utilizar la extensión .cpp o .h en caso de ser un archivo de Header.
	Ejemplos: Graficador.cpp RaicesReales.cpp AreaTriangulo.cpp JuegoAutos.cpp Auto.h Asteroide.h
Identificadores (Nombre de variables, variables de instancia, objetos, matrices, archivos, métodos, parámetros y nombres de constantes)	• Para variables y variables de instancia añadir un prefijo al nombre de la variable de acuerdo con el tipo de dato. Para int usar **i**, para dobule usar **d**, para float usar **f**, para char usar **c**, para string usar **s**, para bool usar **b**, para long usar **l**. Si se trata de un arreglo se agrega **Arr** después del carácter del tipo de dato, y si se trata de un arreglo de 2 dimensiones (matriz) se agrega **Mat** después del carácter del tipo de dato. Si se trata de un apuntador se agrega **p** después del carácter del tipo de dato. • Utilizar la notación camelCase: La primer letra del nombre de la variable debe ser minúscula y las demás palabras deben inicial con mayúscula (prefijoNombreDescriptivo). • Para el nombre de una constante, todas las palabras deben ser mayúsculas, separadas por carácter _ entre cada palabra (prefijoNOMBRE_DESCRIPTIVO). • No deben llevar acento, ni diéresis, ni Ñ. Observa los siguientes ejemplos:

	Prefijo — Tipo de dato — Ejemplo --- b — booleano — bAprobado c — char — cClave d — double — dTarifa f — float — fDistancia i — int — iNumDatos s — string — sTitulo iArr — arreglo de enteros — iArrNumeros dMat — matriz de doubles — dMatTablero letra inicial — objeto clase Tiempo — tHoraInicio de la clase – en caso de 2 clases con la misma letra inicial, usar 3 letras. arrClase — arreglo de objetos Tiempo — arrTiempo - En caso de tener más de un arreglo de la misma clase usar más palabras para ser específicos. Ejemplo: arrTiempoInicio, arrTiempoFin matClase — matriz de objetos Tiempo — matTiempo - En caso de tener más de una matriz de la misma clase usar más palabras para ser específicos. Ejemplo: matTiempoInicio, matTiempoFin. ip — apuntador a entero — ipDato
	Ejemplos: **//Declaración de constantes** `const int iX_MAX = 1;` `const int iX_MIN = -1;` `const double dDELTA = 0.1;` `const char cCLAVE_DEFAULT = 'G';` `const string sNOMBRE_ESTADO = "Nuevo Leon";` **//Declaración de variables** `double dDirY = 0.0;` `float fPeso = 3.5f;` `int iCont = 0;` `bool bComida = false;` `char cLetra = 'A';` `string sPregunta, sOpcion1;` `double dArrMatShininess[] =` `{50.0, 76.0,77.0, 29.0, 52.0};` `string sMatM[iRENG][iCOL];` `double dMatLightAmbient[][4] =` `{ {0.0, 0.0, 1.0, 1.0 },` `{1.0, 1.0, 1.0, 1.0 }` `};` **//Declaración de funciones con parámetros** `void draw3dString (double dX, int iY, char cSt);` `void reshape (int iWidth, int iHeigth);` `void despliega(string sC1, string sArrM[],int iReng);` **//Declaración de archivos** `ofstream archivoSalida;` `ifstream archivoEntrada;`

<table>
<tr><td>Literales numéricas</td><td>En relación a constantes se deben cumplir las siguientes reglas:
Una constante entera sólo especifica su signo cuando sea negativo.
Una constante con decimales, debe contener al menos un dígito antes del punto y al menos un decimal después del punto.</td></tr>
<tr><td></td><td>Ejemplos:
<pre>dPresion = 0.5
dFuerza = 3.0</pre></td></tr>
<tr><td>Orden de los elementos del programa</td><td>Un programa de C++ tendrá el siguiente orden:
Comentario inicial.
Bibliotecas de c++
using namespace std;
Includes que contienen parte del programa
Declaración de variables y constantes globales
Funciones
Función main</td></tr>
<tr><td>Lugar en el que se deben declarar las variables</td><td>Las variables deben declararse al inicio de la función en la que se usan.
En caso de requerirse variables globales, se colocarán entre los includes y las funciones.
Si se requiere declarar variables para el for, se puede declarar dentro del for, o bien al inicio de la función en la que usa, lo que el programador considere más conveniente.</td></tr>
<tr><td></td><td>Ejemplo:
<pre>#include <iostream>
using namespace std;

// Constante Global
const int MAX_PAG = 45;

int calculaSuma (int iArrListaDatos[], int iN)
{
   double dSuma;

   dSuma = 0;
   for (int iCont = 1; iCont <= iN; iCont++)
   {
        ...
   }
   return dSuma;
}</pre></td></tr>
<tr><td>Comentarios</td><td>Comentario de inicio, que contenga nombre y matrícula del autor y breve descripción del propósito del programa y fecha de realización.
Comentario de la función, debe tener una breve descripción de la función y el propósito de cada parámetro.
Comentario para describir una sección de código dentro de una función. Se recomienda que sea lo más breve posible.
Coloca un comentario cuando requieras especificar lo que se hace en una línea o sección de código.
Para las funciones miembro de una clase, escribir un comentario describiendo su propósito, excepto las de acceso, modificación y constructores.</td></tr>
</table>

	Ejemplos: `//` `// Autor: Pedro Picapiedra` `// Matrícula: A01234567` `// Fecha: 16/07/2020` `//` `// Obtener la sumatoria de las fracciones 1/1 a 1/iN`
	`#include <iostream>` `using namespace std;` `// Función que calcula la sumatoria de fracciones` `// El numerador siempre es 1, el denominador es un contador` `// La cantidad de fracciones a sumar` `// es el valor iN que se recibe como parámetro` `void calculaSuma (int iN)` `{` `  ...` `}`
Líneas y Espacios en blanco	• Las líneas de texto deben tener cuando mucho 80 caracteres de longitud. • Cada línea debe contener solamente un estatuto. • Se debe dejar una línea en blanco antes del comentario de cada función. • Se recomienda dejar un renglón en blanco para separar una sección de la función, en este caso incluir un comentario que indique el propósito de la sección. • Se recomienda dejar un renglón en blanco después de la declaración de las variables locales de la función. • Las expresiones deben contener un espacio en blanco entre los operadores y los operandos (iX + iY). • No debe haber espacio en blanco entre un paréntesis y un elemento, ni de un elemento al signo de puntuación correspondiente (void muestraInfo(int iX, int iY)) . • No espacios en blanco entre los corchetes (paréntesis cuadrados) y su contenido (iArrListaDatos[iCont]).
Sangrías	• La indentación se debe hacer con un carácter tab. • Se utilizará indentación para cada nivel de anidamiento en las estructuras de control. • Se debe indentar el contenido de las funciones.
Uso de llaves	• La llave de apertura se colocará alineada a la izquierda en la siguiente línea de la función o el estatuto al que corresponde. • La llave de apertura se coloca sola en una línea. • La llave de cierre se coloca sola en una línea y alineada a su correspondiente llave de apertura. Con excepción de la llave de cierre del do-while que se colocará en el mismo renglón que el while. • Se permitirá no utilizar llaves cuando se tenga una sola instrucción dentro de los estatutos de control (if, while, for).

	Ejemplo: `//` `// Autor: Pedro Picapiedra` `// Matrícula: A01234567` `// Fecha: 16/07/2020` `//` `// Determinar el número más grande de una lista` `//` `#include <iostream>` `using namespace std;`

```
// Leer la cantidad de números que tiene la lista
// Valida que la cantidad sea positiva
int leeCantidad()
{
    int iN;

    // Validar que la cantidad de números sea positiva
    do
    {
        cout << "Cantidad de numeros a leer? ";
        cin >> iN;
    } while (iN <= 0);

    return iN;
}

// Leer los números y determinar cuál es el más grande
int determinaMayor(int iN)
{
    int iMayor, iNum;

    // El primer valor que pide lo toma como el mayor
    cout << "Teclea un numero: ";
    cin >> iMayor;

    // Leer los siguientes n-1 valores y
    // determinar el mayor.
    for (int iCont = 1; iCont <= iN - 1; iCont++)
    {
        cout << "Teclea un numero: ";
        cin >> iNum;

        // Compara y guarda el mayor
        if (iNum > iMayor)
            iMayor = iNum;
    }

    return iMayor;
}

// Función principal
int main()
{
    int iN, iMayor;

    // Leer la cantidad de valores que tecleará el usuaio
    iN = leeCantidad();

    // Determinar el valor más grande de la lista de
    // números tecleados por el usuario
    cout << "Teclea los números" << endl;
    iMayor = determinaMayor(iN);

    // Desplegar el número mayor
    cout << "El número mas grande es: " << iMayor << endl;

    return 0;
}
```

<table>
<tr><td>Nombre de clases o estructuras</td><td>

Utilizar la notación PascalCase: La primer letra en cada palabra debe ser mayúscula (NombreDescriptivo).
No deben llevar acento, ni diéresis ni Ñ.
Si es una clase abstracta, se recomienda que su NombreDescriptivo incluya la palabra Abstract como sufijo.

</td></tr>
<tr><td></td><td>Ejemplos:
MamiferoAbstract
Auto
NaveEspacial</td></tr>
<tr><td>Orden de los elementos dentro de una clase.</td><td>Una clase en C++ tendrá el siguiente orden:

Comentario inicial.
Encabezado de la clase
Sección pública
Sección privada

</td></tr>
<tr><td></td><td>Ejemplo:
<pre>
// Clase Ejemplo
// El propósito de este comentario es describir
// brevemente la clase
class Ejemplo
{
      public:
            // Constructores
            Ejemplo ();
            Ejemplo (int iDato1, double dDato2);

            // Metodos de modificacion
            void setDato1(int iDato1);
            void setDato2(double dDato2);

            // Metodos de acceso
            int getDato1();
            double getDato2();

            // Otros métodos
            void muestra();

      private:
            // Atributos
            int iDato1,
            double dDato2;
};

Ejemplo:: Ejemplo ()
{
      iDato1= 12;
      dDato2= 0.0;
}

Ejemplo:: Ejemplo (int iDato1, double dDato2)
{
      this-> iDato1= iDato1;
      this-> dDato2= dDato2;
}
</pre>
</td></tr>
</table>

```
void Ejemplo::setDato1(int iDato1)
{
      this-> iDato1= iDato1;
}

void Ejemplo::setDato2(double dDato2)
{
      this-> dDato2= dDato2;
}

int Ejemplo::getDato1()
{
      return iDato1;
}

double Ejemplo::getDato2()
{
      return dDato2;
}

void Ejemplo::muestra()
{
      cout << "dato 1 "<< iDato1 << " dato 2 "<< dDato2 <<
endl;
}
```

Parámetros en Constructores

- Los parámetros de los constructores deben llamarse igual que los atributos que se desean inicializar

Ejemplo:

```
class Reloj
{
     public:
         Reloj(int, int);

     private:
         int iHora;
         int iMinutos;
};

Reloj::Reloj(int iHora, int iMinutos)
{
    this->iHora = iHora;
    this->iMinutos = iMinutos;
```

Mejores prácticas

<table>
<tr><td>Salidas de las Funciones</td><td>

- Cuidar que las funciones del programa tengan un return para cada uno de los casos de salida.

</td></tr>
<tr><td></td><td>

Ejemplo:

```
string operacion(int iDia)
{
    switch (iDia)
    {
        case 1:
            return "Domingo";
        case 2:
            return "Lunes";
        case 3:
            return "Martes";
        case 4:
            return "Miercoles";
        case 5:
            return "Jueves";
        case 6:
            return "Viernes";
        case 7:
            return "Sabado";
    }

    return "";
}
```

</td></tr>
<tr><td>Salida de una función booleana</td><td>

- En caso de que la salida de una función sea un valor booleano y éste sea el resultado de una operación, no utilizar una condicional para su salida

</td></tr>
<tr><td></td><td>

Utilizar:

```
bool Reloj::esPM()
{
   return (this->iHora >= 12);
}
```

En lugar de utilizar:

```
bool Reloj::esPM()
{
    if (this->iHora >= 12)
       return true;
    else
       return false;
}
```

</td></tr>
<tr><td>Evitar los estatutos condicionales innecesarios dentro de los ciclos</td><td>

- Revisar la relevancia de los estatutos condicionales dentro de los ciclos, para evitar su operación innecesaria.

</td></tr>
</table>

	Utilizar: `int iTotal = 0;` `for (int iK = 1; iK <= iFin; iK += 2)` `  iTotal += iK;` **En lugar de utilizar:** `int iTotal = 0;` `for (int iK = 1; iK <= iFin; iK++)` `  if (iK % 2 != 0)` `  iTotal += iK;`
Evitar la comparación con una variable booleana	• No realizar comparaciones con variables y/o operaciones que regresen un valor booleano
	Utilizar: `bool bCalcula = true;` `int iTotal = 0;` `...` `if (bCalcula)` `{` `  iTotal += 100;` `}` **En lugar de utilizar:** `bool bCalcula = true;` `int iTotal = 0;` `...` `if (bCalcula == true)` `{` `  iTotal += 100;` `}`
Utilizar la condicional entera	• Se recomienda aprovechar que C++ trabaja con todo lo del lenguaje C, y con ello las condicionales son enteras, es decir 0 significa falso y cualquier valor entero diferente de 0 es verdadero
	Utilizar: `int iTotal = 0;` `...` `if (iTotal)` `  iTotal += 100;` `else` `  iTotal += 50;` **En lugar de utilizar:** `int iTotal = 0;` `...` `if (iTotal != 0)` `  iTotal += 100;` `else` `  iTotal += 50`
Variables globales	• Se recomienda no utilizar variables globales a menos que haya una razón importante por la cual hacerlo.

Ejemplos de programas empleando el estándar de codificación:

```
//
// Autor: Pedro Picapiedra
// Matrícula: A01234567
```

```
// Fecha: 16/07/2020
//
// Calcula la velocidad de un vehículo.

#include <iostream>
using namespace std;

// Calcula la velocidad, dadas la distancia y
// el tiempo que recibe como parámetros
double calcVelocidad (double dDist, double dTime)
{
   double dVel;

   dVel = dDist / dTime;
   return dVel;
}

// Función principal del programa
int main ()
{
   double dDistancia, dVelocidad, dTiempo;

   cout << "Teclea la distancia ";
   cin>>dDistancia;
   cout << "teclea el tiempo ";
   cin>>dTiempo;

   // Si dTiempo es > a 0 llama a la función calcVelocidad
   if (dTiempo <= 0)
       cout << "No se puede calcular la velocidad " << endl;
   else
   {
       dVelocidad = calcVelocidad(dDistancia, dTiempo);
       cout << "La velocidad es " << dVelocidad << endl;
   }

   return 0;
}
```

Bibliography

Free Software Foundation, Inc. (2013, 9 30). *GCC Conding Conventions*. (t. G. Team, Producer) Retrieved 5 28, 2014, from GCC, the GNU Compiler Collection: http://gcc.gnu.org/codingconventions.html#variable

Documento elaborado por:
Comité del Área de Programación, Ciencias Computacionales, Campus Monterrey.
Profesores responsables: Ing. Luis Humberto González Guerra, Ing. Ma. Guadalupe Roque Díaz de León, Ing. Yolanda Martínez Treviño.