

1 But des règles de codage

Le coût de la conception de logiciels est très élevé. Pour préserver ses investissements, l'entreprise sérieuse édite des règles de codage motivées par les raisons suivantes :

- un logiciel doit en tout temps rester modifiable, adaptable et compréhensible ;
- une fonction réalisée doit pouvoir être récupérée sans difficultés pour un nouveau projet ;
- l'indisponibilité d'un programmeur ne doit jamais mettre le projet en péril ;
- des projets réalisés en équipe doivent être cohérents.

Exemple du projet *open source WebKit* :

www.webkit.org/coding/coding-style.html

2 Documentation

Les parties importantes sont expliquées par un organigramme, un diagramme d'état, un structogramme ou du pseudo-code qui peuvent être manuscrits ou réalisés au moyen d'un logiciel spécialisé.

2.1 En-tête du programme

Chaque programme commence par un en-tête qui comprend au minimum les informations suivantes :

- le nom du projet ;
- le nom de l'auteur du programme ;
- la description générale du projet ;
- la date et la référence de la version de base ;
- la date et la référence de chaque version avec la description des modifications.

Exemple :

```
1 /*
2  * Project : RPNCalculator
3  * Author  : GAF
4  * Desc.   : Calculator (4 operations), Reverse Polish Notation
5  * Version : 1.0, 2012.09.14, GAF, initial version
6  */
```

Listing 1 – Entête de programme (*english*)

2.2 Commentaires du programme

Les commentaires sont censés aider la personne qui lira votre programme. Ils ne sont pas là pour indiquer ce que le code dit déjà de façon manifeste.

« *Ne soulignez pas ce qui est évident.* »

Il n'est pas utile de commenter chaque ligne, mais il faut fournir les renseignements clés.

Les éléments suivants doivent *obligatoirement* être commentés :

- les définitions de classes ;
- les déclarations de constantes, d'objets et de variables ;
- les déclarations de méthodes ;
- les lignes de code complexes.

```
1 spaceCount = spaceCount + 1; // increment spaceCount counter
2 total = numberReceived;      // initialize total with ←
   numberReceived
```

Listing 2 – Contre-exemples de commentaires (*english*)

Tous les commentaires du listing 2 sont inutiles et devront être supprimés. Ils n'apportent rien à la compréhension du code. Le lecteur lit couramment le langage *C#*.

2.3 Documentation des méthodes des classes créées

Chaque méthode possède une documentation. Les renseignements permettent à un programmeur étranger au projet d'utiliser la méthode sans devoir analyser son code. L'en-tête comprend au moins :

- la description du but de la méthode (ce qu'elle fait) ;
- la description des paramètres d'entrées et de sortie.

Exemple :

```
1 /*
2 Name : ProfitPerProductPeriod
3 Description : compute profit per product and period
4 Input param. : product reference
5                 start date
6                 end date
7 Output param. : profit
8 */
```

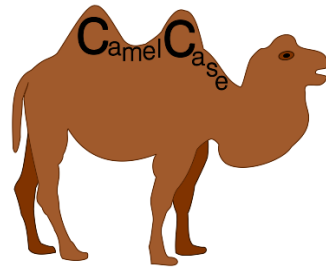
Listing 3 – Entête de méthode (*english*)

3 Identificateurs

Les noms des identificateurs¹ doivent être **auto-explicatifs** et rédigés en **anglais**².

La technique *CamelCase*^a doit être utilisée pour séparer les mots.

^a fr.wikipedia.org/wiki/CamelCase



3.1 Constantes

Le nom des constantes est exprimé entièrement en lettres majuscules en utilisant le caractère *souligné* pour séparer les mots.

```
1 const int NB_COLUMNS_MAX = 75;
```

Listing 4 – Constante (*english*)

3.2 Contrôles *Visual Studio*

Pour faciliter la compréhension, les identificateurs des objets graphiques générés par *Visual Studio* doivent être modifiés :

- la numérotation est remplacée par un nom explicite ;
- le type du contrôle est conservé et doit être abrégé.

Exemple :

```
1 button1_Click;      // C'est le nom généré par Visual Studio
2 buttonStartClick;   // Le nom du contrôle plus sa fonction
3 btnStartClick;      // Abréviation du contrôle plus sa fonction
```

Listing 5 – Identificateurs des contrôles *Visual Studio*



Exception : pour des raisons internes à *Visual Studio* (utilisation du nom de la fiche pour nommer les fichiers), la fiche (**Form1**) ne sera pas renommée.

3.2.1 Abréviations standards

Il est possible d'abrégier les noms des contrôles selon les conventions des tableaux 1(a), 1(b) et 1(c) page 4.

3.3 Classes

Les noms de classes débutent par une majuscule. Ex : **Person**

-
1. classe, type, variable, méthode. . .
 2. voir listing 1 *C#* en grec en annexe A

<i>Nom du contrôle</i>	<i>Abréviation</i>
Button	btn
CheckBox	chk
CheckedListBox	clb
ComboBox	cmb
DateTimePicker	dtp
Label	lbl
LinkLabel	llb
ListBox	lsb
ListView	lsv
MaskedTextBox	mtb
MonthCalendar	mca
NotifyIcon	nic
NumericUpDown	nud
PictureBox	pib
ProgressBar	prb
RadioButton	rdb
RichTextBox	rtb
TextBox	tbx
Timer	tmr
ToolTip	tot
TreeView	tv
WebBrowser	web

(a) Contrôles communs

<i>Nom du contrôle</i>	<i>Abréviation</i>
HScrollBar	hsb
VScrollBar	vsb

(d) Autres contrôles

<i>Nom du contrôle</i>	<i>Abréviation</i>
FlowLayoutPanel	flp
GroupBox	gb
Panel	pnl
SplitContainer	sc
TabControl	tc
TableLayoutPanel	tlp

(b) Conteneurs

<i>Nom du contrôle</i>	<i>Abréviation</i>
ContextMenuStrip	cms
MenuStrip	ms
StatusStrip	ss
ToolStrip	ts
ToolStripContainer	tsc
ToolStripMenuItem	tsm
ColorDialog	cd
FolderBrowserDialog	fbd
FontDialog	fd
OpenFileDialog	ofd
SaveFileDialog	sfd
Status Bar	stb

(c) Menus, barres d'outils
et boîtes de dialogues

TABLE 1 – Abréviations des contrôles

3.4 Objets

Les noms des objets débutent par une minuscule. Ex : `person`

3.5 Champs privés

Les noms des champs privés débutent par le caractère souligné suivi d'une minuscule.
Ex : `private double _rate`

3.6 Propriétés

Les propriétés (en *C#*) sont les méthodes d'accès aux champs privés (`getter/setter`).
Les noms des propriétés débutent par une majuscule. Ex :

```
1 public double Rate {  
2     get { return _rate; }  
3     set { _rate = value; }  
4 }
```

Listing 6 – Propriété (*english*)

3.7 Méthodes

Les noms des méthodes débutent par une majuscule.

Utilisez des verbes actifs, éventuellement suivis par des noms.

Ex :

- InitializeMaze
- ClearMaze
- ComputeMaze
- CalculateInvoiceTotal
- UpdateAddress, MakeStateSelector
- AddObject, ReleaseObject
- AddInterest, ConvertEuroToDollar

Voir liste de verbes d'action en annexe C.

3.8 Exemple complet

```
1 /*  
2  * Project : 02  
3  * Author  : GAF  
4  * Desc.   : Basis of interfaces components  
5  * Version : 1.0, 2012.09.14, GAF, initial version  
6  */  
7  
8 using ...  
9  
10 namespace M120  
11 {  
12     public partial class Form1 : Form  
13     {
```

```

14     private double _rate;
15
16     public double Rate {
17         get { return _rate; }
18         set { _rate = value; }
19     }
20
21     public Form1() {
22         InitializeComponent();
23     }
24
25     private void btnDisplay_Click(object sender, EventArgs e) {
26         // display the text which was read in TextBox tebInput
27         string text = tebInput.Text.Trim();
28         if (text.Length != 0) {
29             this.Rate = Convert.ToDouble(text);
30             MessageBox.Show("Texte saisi= " + this.Rate.ToString(), ←
31                             "Vérification de la saisie", MessageBoxButtons.OK, ←
32                             MessageBoxIcon.Information);
33         }
34         else {
35             MessageBox.Show("Saisissez une valeur...", "Vérification ←
36                             de la saisie", MessageBoxButtons.OK, ←
37                             MessageBoxIcon.Error);
38         }
39     }
40 }

```

Listing 7 – Exemple complet (*english*)

3.9 Référence

msdn.microsoft.com/en-us/library/vstudio/ms229002%28v=vs.100%29.aspx
sites.google.com/site/notionscsharpcem/guiconroles/gui-prefixes

4 Valeurs numériques

Le code ne doit contenir **aucune valeur numérique** à l'exception de 0 et 1. Les autres valeurs sont toutes déclarées sous forme de constantes. Cette méthode facilite la mise à jour et rend les programmes compréhensibles.

Exemple :

```

1 for (int i=0; i < 75; ++i)
2 {
3     if (s > 10)
4     {
5         s = 10;
6     }
7 }

```

Listing 8 – Source avec des valeurs numériques

4 défauts :

- la valeur 75 n'est pas auto-documentée ;
- si la valeur 10 doit être changée, il faudra faire deux modifications ;

- la valeur 10 n'est pas auto-documentée ;
- la variable `s` n'est pas auto-documentée.

```

1 const int NB_COLUMNS_MAX = 75;
2 const int RED_COLOR_MAX = 10;
3 for (int i=0; i < NB_COLUMNS_MAX; ++i)
4 {
5     if (redColorThreshold > RED_COLOR_MAX)
6     {
7         redColorThreshold = RED_COLOR_MAX;
8     }
9 }

```

Listing 9 – Source avec des constantes auto-documentées (*english*)

4 avantages :

- la constante `NB_COLUMNS_MAX` est auto-documentée
- le changement de la limite nécessite une seule modification (`RED_COLOR_MAX`) ;
- la constante `RED_COLOR_MAX` est auto-documentée ;
- la variable globale `GRedColorThreshold` est auto-documentée.

5 Raccourcis claviers

Liste exhaustive : www.dofactory.com/ShortCutKeys/ShortCutKeys.aspx

5.1 Génération/exécution/débogage

F5	démarre le débogage
SHIFT+F5	arrête le débogage
CTRL+F5	exécute sans débogage
F6	génère la solution
F10/F11	exécute un pas (principal/détaillé)

5.2 Complétion, édition

TAB	insère un modèle de code (<code>if</code> , <code>for</code> , <code>foreach</code> , <code>class...</code>) $2 \times$ TAB pour un modèle pré-rempli
CTRL+SPACE	complète le mot (variable, méthode...)
CTRL+L	coupe la ligne courante (<i>Line</i>)
SHIFT+ALT+T	déplace la ligne courante vers le bas (<i>Toggle</i>)
SHIFT+ALT+ARROW	sélectionne en mode bloc
CTRL+I	recherche incrémentale (<i>Incremental</i>)
CTRL+SHIFT+R	enregistrement d'une macro (<i>Record</i>)
CTRL+SHIFT+P	exécution de la macro enregistrée (<i>Play</i>)
Menu contextuel (clic-droit)	<ul style="list-style-type: none"> — recherche le <i>using</i> de la classe — génère le squelette de la méthode — génère les méthodes d'une interface

5.3 Formatage, insertion

CTRL+K, CTRL+F	auto-indent la sélection
CTRL+K, CTRL+D	auto-indent le fichier
CTRL+K, CTRL+C	commente un bloc (<i>Comment</i>)
CTRL+K, CTRL+U	décommente un bloc (<i>Uncomment</i>)
CTRL+K, CTRL+S	entoure la sélection de : #region (<i>Surround</i>)...
CTRL+K, CTRL+X	insère un <i>snippet</i>

5.4 Navigation/fenêtre

F7	affiche la fenêtre de code
SHIFT+F7	affiche la fenêtre d'interface graphique
CTRL+K, CTRL+K	place/retire un signet (<i>bookmark</i>)
CTRL+K, CTRL+N	va au prochain signet (<i>Next</i>)
CTRL+K, CTRL+P	va au précédent signet (<i>Previous</i>)
CTRL+W, D	affiche la fenêtre de définition de code (classe, structure...) (<i>Definition</i>)
CTRL+M, CTRL+M	ouvre/ferme un pli
CTRL+M, CTRL+L	plie/déplie toutes les régions

Annexes

A Source C# en grec

```
02/09/2007 V1
Παρατηρήσεις
- Η επικοινωνία με τον «έξω» κόσμο γίνεται
μέσω της βιβλιοθήκης συναρτήσεων
- Το σημείο έναρξης του προγράμματος είναι η
«κύρια» συνάρτηση main.
*/
namespace WindowsFormsGrec
{
    public partial class Ιδιαίτερα : Form {
        int πραγματικοί = 12;

        public Ιδιαίτερα() {
            InitializeComponent();
        }

        private void Αναθεώρηση_Click(object sender, EventArgs e) {
            MessageBox.Show("Ο ΣΤΔ του μονοδιάστατου πίνακα επαρκεί : " +
                (πραγματικοί++).ToString());
        }
    }
}
```

FIGURE 1 – Form1.cs

Seriez-vous capable d'apporter des modifications au listing de la figure 1 ?

B IOCCC

Exemple de mise en page à ne pas suivre...

IOCCC³ goals :

- to show the importance of programming style, in an ironic way
- to stress C compilers with unusual code
- to illustrate some of the subtleties of the C language
- to provide a safe forum for poor C code. ☺

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int a, char **A){FILE*B; typedef unsigned long C; C b
5 [8]; if(!(a==7&&(B= fopen(1[A], "rb")))) return 1; for(7[b]=0
6 ;7[b]<5;7[b]++)b[7[
7 b]=3[b]
8 )!=(C)-
9 ;7[b]<4
10 b])^(6[
11 <<7[b])
12 <<(0[b]
13 ++))if((
14 [b]=(5[
15 b]<=1;
16 -1))-1)
17 b]=0;7[
18 if(((5[b]>>7[b])^(5
19 1<<7[b])^((C)1<<(0[
20 printf("%0*1X\n", ( int)(0[ b]+3)>> 2,5[b]); return 0;}
```

Listing 10 – tomtorfs.c

The code above is valid and computes CRC32.

(Winner of IOCCC 1998, <http://www.ioccc.org/years.html#1998>)

3. *International Obfuscated C Code Contest*, www.ioccc.org, « Concours international de code C obscur »

C Verbes d'action

Add	Exit	Protect	Start
Approve	Expand	Publish	Step
Assert	Export	Push	Stop
Backup	Find	Read	Submit
Block	Format	Receive	Suspend
Checkpoint	Get	Redo	Switch
Clear	Grant	Register	Sync
Close	Group	Remove	Test
Compare	Hide	Rename	Trace
Complete	Import	Repair	Unblock
Compress	Initialize	Request	Undo
Confirm	Install	Reset	Uninstall
Connect	Invoke	Resolve	Unlock
Convert	Join	Restart	Unprotect
ConvertFrom	Limit	Restore	Unpublish
ConvertTo	Lock	Resume	Unregister
Copy	Measure	Revoke	Update
Debug	Merge	Save	Use
Deny	Mount	Search	Wait
Disable	Move	Select	Watch
Disconnect	New	Send	Write
Dismount	Open	Set	
Edit	Out	Show	
Enable	Ping	Skip	
Enter	Pop	Split	

Remarque : ces verbes d'action proviennent de la commande `Get-Verb` du langage de script de *Windows : PowerShell 2*

Voir msdn.microsoft.com/en-us/library/ms714428.aspx

D Mots réservés

AddHandler	Byte	Char	Decimal
AddressOf	ByVal	CInt	Declare
Alias	Call	Class	Default
And	Case	CLng	Delegate
Ansi	Catch	CObj	Dim
As	CBool	Const	Do
Assembly	CByte	CShort	Double
Auto	CChar	CSng	Each
Base	CDate	CStr	Else
Boolean	CDec	CType	ElseIf
ByRef	Cdbl	Date	End

Enum	Let	Overridable	String
Erase	Lib	Overrides	Structure
Error	Like	ParamArray	Sub
Event	Long	Preserve	SyncLock
Exit	Loop	Private	Then
ExternalSource	Me	Property	Throw
False	Mod	Protected	To
Finalize	Module	Public	True
Finally	MustInherit	RaiseEvent	Try
Float	MustOverride	ReadOnly	TypeOf
For	MyBase	ReDim	Unicode
Friend	MyClass	Region	Until
Function	Namespace	REM	volatile
Get	New	RemoveHandler	When
GetType	Next	Resume	While
Goto	Not	Return	With
Handles	Nothing	Select	WithEvents
If	NotInheritable	Set	WriteOnly
Implements	NotOverridable	Shadows	Xor
Imports	Object	Shared	eval
In	On	Short	extends
Inherits	Option	Single	instanceof
Integer	Optional	Static	package
Interface	Or	Step	var
Is	Overloads	Stop	

Liste des tableaux

1	Abréviations des contrôles	4
---	--------------------------------------	---

Listings

1	Entête de programme (<i>english</i>)	1
2	Contre-exemples de commentaires (<i>english</i>)	2
3	Entête de méthode (<i>english</i>)	2
4	Constante (<i>english</i>)	3
5	Identificateurs des contrôles <i>Visual Studio</i>	3
6	Propriété (<i>english</i>)	5
7	Exemple complet (<i>english</i>)	5
8	Source avec des valeurs numériques	6
9	Source avec des constantes auto-documentées (<i>english</i>)	7
10	<code>tomtorfs.c</code>	10

Table des matières

1	But des règles de codage	1
2	Documentation	1
2.1	En-tête du programme	1
2.2	Commentaires du programme	2
2.3	Documentation des méthodes des classes créées	2
3	Identificateurs	3
3.1	Constantes	3
3.2	Contrôles <i>Visual Studio</i>	3
3.2.1	Abréviations standards	3
3.3	Classes	3
3.4	Objets	5
3.5	Champs privés	5
3.6	Propriétés	5
3.7	Méthodes	5
3.8	Exemple complet	5
3.9	Référence	6
4	Valeurs numériques	6
5	Raccourcis claviers	7
5.1	Génération/exécution/débogage	7
5.2	Complétion, édition	7
5.3	Formatage, insertion	8
5.4	Navigation/fenêtre	8

A	Source <i>C#</i> en grec	9
B	IOCCC	10
C	Verbes d'action	11
D	Mots réservés	11



Ce document est publié par le DIP Genève (CFPT, École d'Informatique) sous licence Creative Commons - utilisation et adaptation autorisées sous conditions.

Auteurs : J.-M. Court, F. Garcia et C. Maréchal

www.ge.ch/sem/cc/by-nc-sa