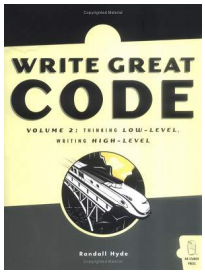


1 But des règles de codage

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 de fichier

Chaque fichier 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 :

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

Listing 1 – Entête de fichier (*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.

```
spaceCount = spaceCount + 1; // increment spaceCount counter
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 :

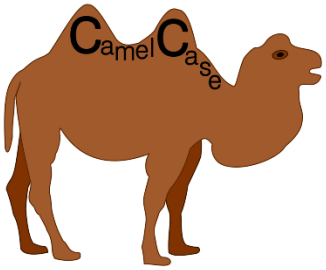
```
/*
Name : ProfitPerProductPeriod
Description : compute profit per product and period
Input param. : product reference
               start date
               end date
Output param. : profit
*/
```

Listing 3 – Entête de méthode (*english*)

3 Identificateurs

Les noms des identificateurs [1] 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.

```
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 :

```
button1_Click;      // C'est le nom généré par Visual Studio
buttonStartClick;   // Le nom du contrôle plus sa fonction
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'abréger 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

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

(a) Contrôles communs

Nom du contrôle	*Abréviation*
FlowLayoutPanel	flp
GroupBox	gb
Panel	pnl
SplitContainer	sc
TabControl	tc
TableLayoutPanel	tlp

(b) Conteneurs

Nom du contrôle	*Abréviation*
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

Nom du contrôle	*Abréviation*
HScrollBar	hsb
VScrollBar	vsb

(d) Autres contrôles

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 :

```
public double Rate {
  get { return _rate; }
  set { _rate = value; }
}
```

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

```
/*
 * Project : 02
 * Author  : GAF
 * Desc.   : Basis of interfaces components
 * Version : 1.0, 2012.09.14, GAF, initial version
 */

using ...

namespace M120
{
  public partial class Form1 : Form
  {
    private double _rate;

```

```
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(), ↩
              "Vérification de la saisie", MessageBoxButtons.OK, ↩
              MessageBoxIcon.Information);
31        }
32        else {
33          MessageBox.Show("Saisissez une valeur...", "Vérification ↩
              de la saisie", MessageBoxButtons.OK, ↩
              MessageBoxIcon.Error);
34        }
35      }
36    }
37  }
```

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/guicontroles/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.

```
const int NB_COLUMNS_MAX = 75;
const int RED_COLOR_MAX = 10;
for (int i=0; i < NB_COLUMNS_MAX; ++i)
{
  if (redColorThreshold > RED_COLOR_MAX)
  {
    redColorThreshold = RED_COLOR_MAX;
  }
}
```

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`	génére la solution
`F10/F11`	exécute un pas (principal/détaillé)

5.2 Complétion, édition

`TAB`	insére un modèle de code (`if, for, foreach, class...`) 2 × `TAB` pour un modèle pré-rempli
`CTRL+SPACE`	complète le mot (variable, méthode...)
`CTRL+L`	coupe la ligne courante (*Line*)
`SHIFT+ALT+T`	déplace la ligne courante vers le bas (*Toggle*)
`SHIFT+ALT+ARROW`	sélectionne en mode bloc
`CTRL+I`	recherche incrémentale (*Incremental*)
`CTRL+SHIFT+R`	enregistrement d'une macro (*Record*)
`CTRL+SHIFT+P`	exécution de la macro enregistrée (*Play*)
Menu contextuel (clic-droit)	— recherche le *using* de la classe — génère le squelette de la méthode — génère les méthodes d'une interface

5.3 Formatage, insertion

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

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 (*booKmark*)
`CTRL+K, CTRL+N`	va au prochain signet (*Next*)
`CTRL+K, CTRL+P`	va au précédent signet (*Previous*)
`CTRL+W, D`	affiche la fenêtre de définition de code (classe, structure...) (*Definition*)
`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...

IOCCC[3] 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. ☺

```
#include <stdio.h>
#include <stdlib.h>

int main(int a,char      **A){FILE*B;typedef      unsigned long C;C b
[8]; if(!(a==7&&(B=      fopen(1[A],"rb"))))      return 1;for(7[b]=0
;7[b]<5;7[b]++)b[7[      b]]=strtoul(A[2+7[b      ]],0,16-!7[b]*6);5[
b]=3[b]                  ; while      ((6[b]=     getc(B)
)!=(C)-                  1){if(2      [b])for     (7[b]=0
;7[b]<4                  ;7[b]++      )if(((6     [b]>>7[
b])^(6[                  b]>>(7-7[b])))&1)6[      b] ^=(1
<<7[b])                  ^(1<<(7-7[b]));5[b]      ^= 6[b]
<<(0[b]                  -8);for(7[b]=0;7[b]      <8;7[b]
++)if((                  5[b]>>(0[b]-             1))&1)5
[b]=(5[                  b]<<1)^ 1[b];            else 5[
b]<<=1;                  }5[b]&=((((C)1           <<(0[b]
-1))-1)                  <<1)|1; if(2[b]          )for(7[
b]=0;7[                  b]<(0[b  ]>>1);7         [b] ++)
if(((5[b]>>7[b])^(5      [b]>>(0   [b]-1-7        [b])))&1)5[b]^=((C)
1<<7[b])^((C)1<<(0[      b]-1-7[    b]));5[       b]^=4[b];fclose(B);
printf("%0*lX\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	Enter	Protect	Split
Approve	Exit	Publish	Start
Assert	Expand	Push	Step
Backup	Export	Read	Stop
Block	Find	Receive	Submit
Checkpoint	Format	Redo	Suspend
Clear	Get	Register	Switch
Close	Grant	Remove	Sync
Compare	Group	Rename	Test
Complete	Hide	Repair	Trace
Compress	Import	Request	Unblock
Confirm	Initialize	Reset	Undo
Connect	Install	Resolve	Uninstall
Convert	Invoke	Restart	Unlock
ConvertFrom	Join	Restore	Unprotect
ConvertTo	Limit	Resume	Unpublish
Copy	Lock	Revoke	Unregister
Debug	Measure	Save	Update
Deny	Merge	Search	Use
Disable	Mount	Select	Wait
Disconnect	Move	Send	Watch
Dismount	Open	Set	Write
Edit	Ping	Show	
Enable	Pop	Skip	

Remarque : ces verbes d'action proviennent de la commande `Get-Verb` du langage de script de *Windows* : *PowerShell 2*
Voir `https://docs.microsoft.com/en-us/powershell/developer/cmdlet/approved-verbs-for-windows-powershell-commands`

D Mots réservés

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/

abstract	event	new	struct
as	explicit	null	switch
base	extern	object	this
bool	false	operator	throw
break	finally	out	true
byte	fixed	override	try
case	float	params	typeof
catch	for	private	uint
char	foreach	protected	ulong
checked	goto	public	unchecked
class	if	readonly	unsafe
const	implicit	ref	ushort
continue	in	return	using
decimal	int	sbyte	usingstatic
default	interface	sealed	virtual
delegate	internal	short	void
do	is	sizeof	volatile
double	lock	stackalloc	while
else	long	static	
enum	namespace	string	

Liste des tableaux

1 Abréviations des contrôles 4

Listings

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

Table des matières

1 But des règles de codage **1**

2 Documentation **1**
2.1 En-tête de fichier 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 *Visual Studio* 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 *C#* en grec **9**

B IOCCC **10**

C Verbes d'action **11**

D Mots réservés **12**