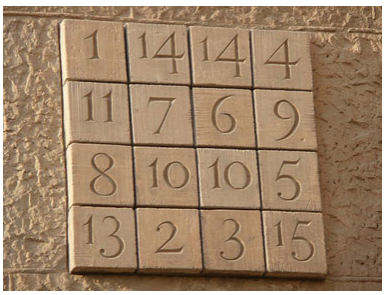
Projet Semaine bloquée M1103

Le carré « magique » de la Sagrada Familia

Cryptogramme de Subirachs

©Athos99 M.Bobillier



Le cryptogramme sculpté par **Subirachs**, vu sur la facade (*ci-dessus*) ou sur la porte d'entrée (*ci-contre*) de la cathédrale Sagrada Familia à Barcelone, immense projet de l'architecte **Gaudi**.
(à voir : http://fr.wikipedia.org/wiki/Sagrada_Fam%C3%ADlia)
Ci-dessous, une réalisation sous Excel qui permet de dévoiler toutes les combinaisons de cases du cryptogramme dont la somme fait 33…

Présentation du sujet

- Le cryptogramme de « *Gaudi-Subirachs* » représente des entiers compris entre 1 et 15 répartis sur une grille de 4 cases par 4. Sa particularité est que : quelle que soit la colonne, la ligne, la diagonale, ou encore bien d'autres combinaisons de 4 cases que l'on peut former avec cette grille, la somme des nombres inscrits sur ces cases est égale à … **33**[1]. On compte **88** combinaisons de **4 cases** dont la somme vaut **33** et mieux encore parmi celles-ci, exactement **33** combinaisons de cases contigues (par un côté ou un angle) !!!.
  Si l'on considère aussi les combinaisons de 3, 5, 6 ou 7 cases, on obtient en tout 310 combinaisons dont la somme vaut 33.
- Votre projet doit aboutir à la production de deux programmes, le premier que l'on nommera **gauditext.adb** vous permettra de tester avec une interface classique (vue texte) l'ensemble de vos fonctions et procédures. Le second, **gaudigraph.adb** affichera le carré magique sous forme graphique et réagira dynamiquement aux actions de l'utilisateur.
- Le cryptogramme sera traité comme suit :

	A	B	C	D
1	1	14	14	4
2	11	7	6	9
3	8	10	10	5
4	13	2	3	15

- ✓ Les lignes de la grille sont identifiées par un entier de 1 à 4
- ✓ Les colonnes de la grille sont identifiées par une lettre majuscule de A à D
- ✓ La case située à l'intersection de la troisième colonne et de la troisième ligne sera ainsi identifiée par le couple (C,3) et nommée "C3"

- Les 16 cases de la grille sont stockées dans un fichier séquentiel binaire de nom « **CarreGaudi** ». Ce fichier (disponible dans **/users/info/pub/1a/M1103/SBAda/)** sera à copier dans un vecteur pour chercher les différentes combinaisons et pour afficher la grille dans les différentes vues que vous aurez à créer.

[1] *Age de la crucifixion du Christ selon la religion Chrétienne*

Partie 1 – Combinaisons gagnantes : dites 33 !

L'objectif de cette partie est de déterminer toutes les combinaisons « gagnantes », c'est-à-dire les suites de 3, 4, 5, 6 ou 7 cases dont la somme des valeurs est égale à 33 (*une case n'apparaissant qu'une seule fois dans une combinaison donnée*).

ATTENTION : contrairement aux « carrés magiques » classiques, les combinaisons peuvent être constituées de cases non adjacentes (*par exemple : A1 + D1 + A4 + D4 = 33*)…

Vous disposez de la spécification du paquetage **p_combinaisons** décrit ci-dessous :

```
with sequential_iO;
with text_io; use text_io;

package p_combinaisons is

---- TYPES pour les cases de la grille -------------------------------------------------
  subtype T_Col is character range 'A'..'D';
  subtype T_Lig is positive range 1..4;
  subtype T_nbcases is positive range 3..7;
  type TR_Case is record
     nom      :  string(1..2);  -- nom de la case (ex : "A2")
     valeur   :  positive;      -- nombre porté par la case (ex : 14)
  end record;

---- Instanciation de sequential_IO pour le fichier de description de la grille ------
  package p_Cases_io is new sequential_io (TR_Case); use p_Cases_io;

---- Type pour le vecteur de "travail" -------------------------------------------------
  type TV_Gaudi is array (positive range <>) of TR_Case;

---- Recherche et affichage des combinaisons -------------------------------------------

  procedure CreeVectGaudi(f : in out p_cases_io.file_type; V : out TV_Gaudi);
  -- {f ouvert, V de taille suffisante} => {le contenu de f a été copié dans V}

  procedure TriVectGaudi(V : in out TV_Gaudi);
  -- {} => {V est trié par nom de case}

  procedure CreeFicSol(V : in TV_Gaudi; fsol : in out text_io.file_type);
  -- {fsol ouvert en écriture, V trié par nom de case}
  -- => {fsol contient toutes les combinaisons gagnantes et est structuré selon le format défini (cf. sujet)}

  function NbCombi(fsol : in text_io.file_type; nbcases : in T_nbcases) return natural;
  -- {fsol ouvert, f- = <>}=> {résultat = nombre de combinaisons en nbcases dans fsol}

  function Combi(fsol : in text_io.file_type; nbcases : in T_nbcases; numsol : in positive) return string;
  -- {f ouvert, f- = <>}
  -- => {résultat = chaîne représentant la solution numsol lue dans fsol pour une combinaison de nbcases}

end p_combinaisons;
```

Vous disposez également d'un fichier binaire nommé **CarreGaudi** qui contient les éléments de type TR_Case nécesaires à la construction du carré magique - *exemple d'élément : ("A4", 13)*

Travail à faire :

- Recopiez dans votre répertoire les fichiers du répertoire **`/users/info/pub/1a/M1103/SBAda/`**
- Écrivez et testez au fur et à mesure les fonctions et procédures du paquetage :
  - ✓ Chargement du contenu du fichier binaire **CarreGaudi** dans un vecteur (de 16 éléments)
  - ✓ Tri du vecteur par nom de case croissant
  - ✓ Création d'un **fichier texte** structuré comme indiqué en page suivante, contenant les combinaisons « gagnantes » à 3 cases, 4 cases, 5 cases, 6 cases puis 7 cases
  - ✓ Pour chaque nombre de cases (3, 4, 5, 6 ou 7), affichage des combinaisons de façon lisible (*cf. spécification en page suivante*).
- Testez le paquetage dans un programme de test **`TestCombinaisons.adb`**

REMARQUES : *N'hésitez pas à ajouter d'autres fonctions ou procédures au paquetage si elles vous semblent nécessaires.*
Pensez également à traiter localement et globalement les exceptions **pouvant** *être déclenchées lors de la manipulation d'un fichier (binaire ou texte) et à contrôler les saisies de l'utilisateur.*

Spécification de l'affichage des combinaisons

Affichage des solutions en 7 cases

```
* 8 solutions en 7 cases
-----------------------
solution  1/ 8 : A1,A2,B2,B4,C4,D1,D3
solution  2/ 8 : A1,A3,B2,B4,C2,D1,D3
solution  3/ 8 : A1,A3,B3,B4,C4,D1,D3
solution  4/ 8 : A1,A3,B4,C2,C4,D1,D2
solution  5/ 8 : A1,A3,B4,C3,C4,D1,D3
solution  6/ 8 : A1,B2,B3,B4,C2,C4,D1
solution  7/ 8 : A1,B2,B4,C2,C3,C4,D1
solution  8/ 8 : A1,B2,B4,C2,C4,D2,D3
```

REMARQUE : *Le numéro de chaque solution dépend de la façon dont vous avez programmé la recherche des combinaisons*

Spécification du stockage des solutions suite à appel de CreeFicSol

Lignes relatives aux solutions en 7 cases dans le fichier texte des combinaisons (paramètre fsol de la procédure)

```
7 8
A1A2B2B4C4D1D3
A1A3B2B4C2D1D3
A1A3B3B4C4D1D3
A1A3B4C2C4D1D2
A1A3B4C3C4D1D3
A1B2B3B4C2C4D1
A1B2B4C2C3C4D1
A1B2B4C2C4D2D3
```

← la première ligne ajoutée contient le nombre de cases (7), un espace, puis le nombre de solutions (8)

chaque ligne suivante correspond à une solution représentée par le nom des cases correspondantes
les noms des cases sont accolés les uns aux autres

REMARQUE : *L'ordre d'écriture des combinaisons dépend de la façon dont vous avez programmé la recherche des combinaisons.*

Partie 2 – Vue texte : affichage de la grille et choix des combinaisons

L'objectif de cette partie est de fournir une première interface « console » d'affichage de la grille, à partir de laquelle l'utilisateur pourra choisir le nombre de cases pour lesquelles il veut connaître les solutions (3, 4, 5, 6 ou 7 cases) jusqu'à ce qu'il décide de terminer l'exécution du programme.

Travail à faire :

1. Créez un nouveau paquetage **`p_vue_text`** qui ne sera utilisé qui contiendra les déclarations et les fonctions ou procédures nécessaires à cette partie.
2. Créez une procédure principale **`gauditext.adb`** qui devra (en boucle) :
   - ✓ Demander à l'utilisateur de choisir un nombre de cases
   - ✓ Afficher la grille d'origine
   - ✓ Afficher le nombre de combinaisons pour ce nombre de cases et la liste de ces combinaisons

Exemple d'affichage de la grille :

```
    A     B     C     D
 *-----*-----*-----*-----*
 |     |     |     |     |
1|  1  |  14 |  14 |  4  |
 |     |     |     |     |
 *-----*-----*-----*-----*
 |     |     |     |     |
2|  11 |  7  |  6  |  9  |
 |     |     |     |     |
 *-----*-----*-----*-----*
 |     |     |     |     |
3|  8  |  10 |  10 |  5  |
 |     |     |     |     |
 *-----*-----*-----*-----*
 |     |     |     |     |
4|  13 |  2  |  3  |  15 |
 |     |     |     |     |
 *-----*-----*-----*-----*
```

NOTE :

Ne cherchez pas à afficher dans la console une grille présentant chaque solution pour un nombre de cases donné.
Vous aurez l'occasion de le faire dans une interface graphique nettement plus conviviale..

Si vous en avez le temps, vous pourrez réaliser cette affichage en mode console à titre d'EXTENSION.

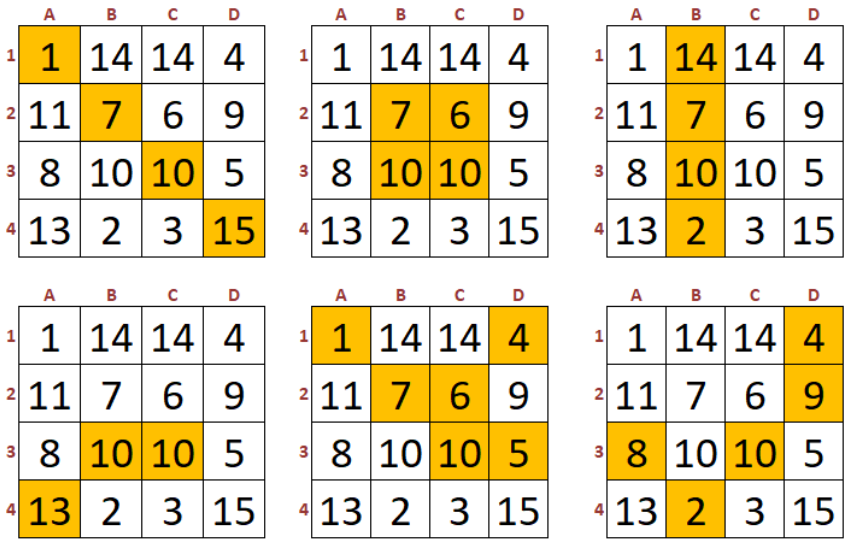
Partie 3 – Solutions contigües

REMARQUE : cette partie peut être développée après la partie 4 qui traite de la vue graphique

- ***On dira que deux cases sont contigües si elles ont au moins un sommet ou un côté commun.***
- ***On dira qu'une combinaison gagnante est contigüe si chaque case de cette combinaison est contigüe avec au moins une autre case de cette même combinaison et que l'on peut « tracer un chemin » d'une case à l'autre de la combinaison.***

L'objectif de cette partie est de déterminer pour chaque nombre de cases (3, 4, 5, 6 ou 7) les combinaisons gagnantes qui sont « contigües » et de les stocker dans un fichier texte de même structure que le fichier des combinaisons précédemment créé par les appels successifs de la procédure **`Combinaisons`**.

EXEMPLES DE COMBINAISONS CONTIGÜES :



INFORMATION :

- ✓ **3** combinaisons contigües de 3 cases
- ✓ **33** combinaisons contigües de 4 cases
- ✓ **35** combinaisons contigües de 5 cases
- ✓ **18** combinaisons contigües de 6 cases
- ✓ **4** combinaisons contigües de 7 cases

Travail à faire :

- Compléter le paquetage **`p_combinaisons`** avec les fonctions ou procédures nécessaires pour déterminer si une combinaison gagnante (somme = 33) est constituée ou non de cases « contigües » et pour stocker toutes les solutions contigües dans un fichier texte structuré comme le fichier des solutions précédemment créé.
- Compléter le programme **`gauditext.adb`** pour permettre à l'utilisateur de n'afficher (s'il le désire) que les combinaisons gagnantes contigües à 3, 4, 5, 6 ou 7 cases.

Partie 4 – Interface graphique

Vous avez pu ressentir une certaine frustration dans les parties prédédentes… Ce serait tellement plus riche de pouvoir révéler les combinaisons gagnantes sur la grille !!!

Pour réaliser cette interface, nous vous proposons un paquetage public **p_fenbase** (voir **Annexe I**).
Ce paquetage permet, **après l'appel d'`InitialiserFenetres`**, de réaliser les opérations décrites ci-dessous :

- **Créer une fenêtre** :

  Une fenêtre a un nom unique et est composée d'un ensemble d'éléments graphiques (boutons rectangulaires, boutons ronds, boutons avec des images dessus (au format .xpm), boîtes à cocher (check box), textes "fixes" (non modifiables par l'utilisateur de l'interface), champs de saisie (modifiables eux !), textes "fixes" avec ascenseur (pour afficher un texte de plusieurs lignes) et horloges (qui affichent l'heure !)).

  - ✓ La création d'une fenêtre commence par l'appel à la fonction **`DebutFenetre`**.
  - ✓ L'ajout d'un élément se fait à l'aide de la procédure relative à son type (**`AjouterBouton`**, **`AjouterBoutonRond`**, **`AjouterTexte`**, **`AjouterChamp`**, **`AjouterTexteAscenseur`**, **`AjouterHorloge`**, **`AjouterBoiteCocher`**, **`AjouterBoutonImage`**).
    - *un élément doit avoir un nom unique pour sa fenêtre. C'est grâce à ce nom qu'on peut le manipuler.*
    - *un élément a une apparence graphique par défaut (ex : fond gris).*
    - *un élément a un comportement par défaut (actif).*
    - *on peut modifier l'apparence et le comportement d'un élément dès sa création (cf. Modifier l'apparence et le comportement d'un élément).*
    - *un élément est défini une fois pour toutes : on ne peut pas le supprimer de sa fenêtre mais on peut le rendre temporairement visible ou invisible (cf. Montrer/Cacher un élément)*
  - ✓ La création d'une fenêtre se termine par un appel à la procédure **`FinFenetre`**.

  Attention : il faut construire les fenêtres d'une application indépendemment les unes des autres.

- **Afficher** / **masquer une fenêtre** :

  Une fenêtre peut être affichée ou masquée à l'utilisateur (procédures **`MontrerFenetre`** / **`CacherFenetre`**).

- **Modifier l'apparence et le comportement d'un élément d'une fenêtre** (dès sa création ou au cours de l'exécution) :
  - ✓ **`ChangerTexte`** permet de modifier le texte associé à un élément (par exemple le texte affiché dans un bouton).
  - ✓ **`ChangerContenu`** permet de modifier le contenu d'un champ de saisie ou d'une zone texte avec ascenseur.
  - ✓ **`ChangerCouleurTexte`** et **`ChangerCouleurFond`** permettent de modifier la couleur du contenu (texte) ou du fond d'un élément .
  - ✓ **`ChangerStyleTexte`** et **`ChangerTailleTexte`** permettent de modifier respectivement le style (*standard, gras, italique,…*) et la taille (*standard, moyen, grand, très grand*) d'un texte de bouton, de champ texte ou de zone de saisie.
  - ✓ **`ChangerEtatBouton`** permet d'activer/désactiver un élément (quel que soit son type) ; un élément désactivé est toujours affiché mais si l'on clique dessus, c'est sans effet….
  - ✓ **`ChangerImageBouton`** permet de modifier l'image d'un bouton « image ».
  - ✓ **`MontrerElem`**, **`CacherElem`** permettent respectivement de montrer un élément préalablement invisible ou de cacher un élément jusque-là visible (par défaut tous les éléments ajoutés à une fenêtre sont visibles).

- **Exploiter une fenêtre :**

  Chaque élément d'une fenêtre a sa raison d'être ; il peut être purement décoratif (dans ce cas, il faut penser à le rendre inactif) ou fonctionnel (ex : masquer la fenêtre lorsqu'on clique dessus).

  - ✓ La fonction **`AttendreBouton`** bloque l'exécution jusqu'à ce que l'utilisateur ait cliqué sur un élément actif de la fenêtre courante. Cette fonction a pour résultat le nom de l'élément graphique choisi (bouton ou champ de saisie).
  - ✓ La fonction **`ClickDroit`** permet de savoir si l'utilisateur a utilisé le bouton droit de la souris (on peut ainsi effectuer un traitement différencié suivant la nature du click).
  - ✓ La fonction **`ConsulterContenu`** permet de récupérer ce qu'a tapé l'utilisateur dans un élément de type champ de saisie.
  - ✓ La fonction EtatBoiteCocher permet de savoir si une boite à cocher est activée ou non.

  Attention : pour qu'une fenêtre reste affichée après l'appel de **`MontrerFenetre`**, le programme doit comporter au moins un appel à **`AttendreBouton`**.

Pour vous guider dans l'utilisation du paquetage **p_fenbase**, nous vous fournissons le code d'un petit jeu « stupide » : **`marketing.adb`**. Vous pouvez étudier et exécuter ce programme (voir **Annexe II**)…
Certaines de ses astuces peuvent vous être précieuses pour développer votre programme.
En particulier, vous pouvez remarquer que dans la boucle principale de jeu, la déclaration dynamique de la variable **Bouton** permet d'effectuer un seul appel à **`AttendreBouton`** et d'éviter ainsi d'avoir à cliquer plusieurs fois …
(*NOTE : Vous pouvez faire nettement mieux en termes de lisibilité et réutilisabilité de votre code !)*

Travail à faire :

- Créez un nouveau paquetage **`p_vue_graph qui`** contiendra toutes les fonctions et/ou procédures nécessaires à la gestion de la vue graphique ;
- Créez un nouveau programme **`gaudigraph.adb`** qui utilisera ce paquetage ainsi que tous ceux qui lui seront utiles pour offrir (*entre autres*) les fonctionnalités suivantes :
  - ✓ Affichage d'une fenêtre d'accueil
  - ✓ Affichage d'une fenêtre dans laquelle sera représenté le cryptogramme (*sans numérotation des lignes et des colonnes*) ;
  - ✓ Messages d'instructions à l'utilisateur pour le guider dans les choix qu'il peut faire (ex : choisir le nombre de cases des combinaisons, ne voir que les combinaisons contigües, passer à la solution suivante, quitter le programme, etc.) ;
  - ✓ Messages indiquant à l'utilisateur le nombre de combinaisons pour un nombre de cases donné, ou encore le numéro de la combinaison actuellement révélée à l'écran ;
  - ✓ Boutons d'action ou zones de saisie permettant à l'utilisateur de « jouer » avec le cryptogramme conformément aux instructions qui lui sont données ;
  - ✓ Fermeture de la fenêtre quand l'utilisateur décide de quitter le programme.

Les combinaisons gagnantes devront être dévoilées une par une (sur action de l'utilisateur pour passer à la combinaison suivante)

RAPPEL : pour compiler un programme utilisant les bibliothèques graphiques, utilisez la commande **`build`**.

Faites en sorte que votre code soit lisible !!!
Bien entendu, gaudigraph.adb tirera partie des développements que vous avez faits pour chercher et enregistrer les combinaisons gagnantes … Il ne s'agit ici que de travailler l'interface avec l'utilisateur !!!

EXTENSIONS (uniquement si les parties précédentes ont été traitées)

Vous pouvez laisser libre cours à votre imagination pour étendre les fonctionnalités de vos programmes.

À titre d'exemple, vous pourriez faire en sorte que l'on puisse **vraiment** jouer avec le pictogramme de Gaudi :

- *Poser le défi à l'utilisateur de dévoiler le plus possible de solutions en NB cases en un temps limite…*
- *Choisir 2 cases au hasard et demander à l'utilisateur de trouver le plus possible de solutions contigües qui utilisent ces cases (s'il y en a) …*
- *Gérer un historique des scores réalisés par les utilisateurs dans les différentes formes de « jeu avec le pictogramme » que vous aurez développées…*

Vous pouvez aussi étendre l'interface « console » de façon à ce que les solutions en NB cases soient affichées dans la grille, ou encore offrir à l'utilisateur un petit moment culturel en lui donnant accès à une fenêtre d'information sur le cryptogramme de Gaudi- Subirachs.

Annexe I - Spécification du paquetage p_fenbase

```
with Forms; use Forms; with Interfaces.C;use Interfaces.C;
package p_fenbase is
 ------------------------------------------------------------------------------
 -- types à connaître pour utiliser le paquetage
 type TR_Fenetre; -- une fenêtre de l'interface graphique
 type T_EtatBouton is (Marche, Arret); -- état d'un bouton (actif ou inactif)
 subtype T_Couleur is FL_COLOR; -- couleurs disponibles dans le paquetage forms
 ------------------------------------------------------------------------------
 -- types internes au paquetage, inutile de les connaître !
 type TA_String is access String; -- pointeur sur une chaîne de caractères
 type T_TypeElement is (Bouton, TexteFixe, ChampDeSaisie, TexteAscenseur, Horloge, Fond, CheckBox, PictBouton);
 type TR_Element;
 type TA_Element is access TR_Element;
 type TR_Element is record
   TypeElement : T_TypeElement;
   NomElement : TA_String;
   Texte : TA_String;
   Contenu : TA_String;
   PElement : FL_OBJECT_Access;
   Suivant : TA_Element;
 end record;
 type TR_Fenetre is record
   PFenetre : FL_FORM_Access;
   Titre : TA_String;
   PElements : TA_Element;
 end record;
 ------------------------------------------------------------------------------
 -- primitives de la bibliothèque "graphique"
 ------------------------------------------------------------------------------
 procedure InitialiserFenetres;        --    Initialiser le mode graphique
 ------------------------------------------------------------------------------
 function DebutFenetre (               --    Créer une nouvelle fenêtre
   Titre : in String;                  --    son titre
   Largeur,                            --    sa largeur en pixels
   Hauteur : in Positive )             --    sa hauteur en pixels
 return TR_Fenetre;                    --    résultat : la fenêtre créée
 ------------------------------------------------------------------------------
 procedure AjouterBouton (             --    Ajouter un Bouton
   F : in out TR_Fenetre;              --    la fenêtre où on ajoute
   NomElement : in String;             --    le nom du bouton (unique)
   Texte : in String;                  --    le texte affiché dans le bouton
   X, Y : in Natural;                  --    ses coordonnées en pixels
   Largeur, Hauteur : in Positive );   --    largeur et hauteur en pixels
 ------------------------------------------------------------------------------
 procedure AjouterBoutonRond (         --    Ajouter un Bouton de forme ronde
    F : in out TR_Fenetre;             --    la fenêtre où on ajoute
   NomElement : in String;             --    le nom du bouton (unique)
   Texte : in String;                  --    le texte affiché dans le bouton
   X, Y : in Natural;                  --    ses coordonnées en pixels
   Largeur, Hauteur : in Positive );   --    largeur et hauteur en pixels
 ------------------------------------------------------------------------------
 procedure AjouterTexte (              --    Ajouter un texte non modifiable
   F : in out TR_Fenetre;              --    la fenêtre où on ajoute
   NomElement : in String;             --    le nom du bouton (unique)
   Texte : in String;                  --    le texte qui sera affiché
   X, Y: in Natural;                   --    ses coordonnées en pixels
   Largeur, Hauteur : in Positive );   --    largeur et hauteur en pixels
 ------------------------------------------------------------------------------
 procedure AjouterChamp (              --    Ajouter un champ de saisie
   F : in out TR_Fenetre;              --    la fenêtre où on ajoute
   NomElement : in String;             --    le nom du bouton (unique)
   Texte : in String;                  --    le texte affiché en légende
   Contenu : in String;                --    le contenu initial du champ
   X, Y: in Natural;                   --    ses coordonnées en pixels
   Largeur, Hauteur : in Positive );   --    largeur et hauteur en pixels
 ------------------------------------------------------------------------------
 procedure AjouterTexteAscenseur (     --    Ajouter une zone texte avec
   F : in out TR_Fenetre;              --    ascenseur (plusieurs lignes)
   NomElement : in String;             --    le nom du bouton (unique)
   Texte : in String;                  --    le texte affiché en légende
   Contenu : in String;                --    le contenu de la zone
   X, Y : in Natural;                  --    ses coordonnées en pixels
   Largeur, Hauteur : in Positive );   --    largeur et hauteur en pixels
 ------------------------------------------------------------------------------
 procedure AjouterHorloge (            --    Ajouter une horloge qui affiche
   F : in out TR_Fenetre;              --    l'heure courante dans F
   NomElement : in String;             --    le nom du bouton (unique)
   Texte : in String;                  --    le texte affiché en légende
   X, Y : in Natural;                  --    ses coordonnées en pixels
   Largeur, Hauteur  : in Positive ); --     largeur et hauteur en pixels
------------------------------------------------------------------------------
```

```
 procedure AjouterBoiteCocher (        --    Ajouter une boîte à cocher
   F : in out TR_Fenetre;              --    la fenêtre où on ajoute
   NomElement : in String;             --    le nom du bouton (unique)
   Texte : in String;                  --    le texte affiché en légende
   X, Y : in Natural;                  --    ses coordonnées en pixels
   Largeur, Hauteur : in Positive );   --    largeur et hauteur en pixels
 ------------------------------------------------------------------------------
 procedure AjouterBoutonImage (        --    Ajouter un bouton avec image
   F : in out TR_Fenetre;              --    la fenêtre où on ajoute
   NomElement : in String;             --    le nom du bouton (unique)
   Texte : in String;                  --    le texte affiché en légende
   X, Y : in Natural;                  --    ses coordonnées en pixels
   Largeur, Hauteur : in Positive );   --    largeur et hauteur en pixels
 ------------------------------------------------------------------------------
 procedure FinFenetre (                --    Procedure à appeler quand on a
   F : in TR_Fenetre );                --    fini d'ajouter des éléments
 ------------------------------------------------------------------------------
 procedure MontrerFenetre (            --    Afficher unefenêtre
   F : in TR_Fenetre );
 ------------------------------------------------------------------------------
 procedure CacherFenetre (             --    Masquer une fenêtre
   F : in TR_Fenetre );
 ------------------------------------------------------------------------------
 function AttendreBouton (             --    Attendre qu'un bouton soit pressé
   F : in TR_Fenetre )                 --    dans la fenetre F
 return String;                        --    résultat = nom du bouton pressé
------------------------------------------------------------------------------
 function ClickDroit return boolean;   --    vrai si on a pressé le bouton droit de la souris
 ------------------------------------------------------------------------------
 function ConsulterContenu (           --    Permet de récupérer la valeur saisie
   F : in TR_Fenetre;                  --    par l'utilisateur dans un champ de F
   NomElement : in String )            --    le nom du champ de saisie (unique)
   return String;                      --    résultat : la chaîne saisie !
 ------------------------------------------------------------------------------
 function EtatBoiteCocher (            --    Permet de savoir si une Checkbox est
   F : in TR_Fenetre;                  --    cochée ou non
   NomElement : in String )            --    le nom du bouton (unique)
   return boolean;                     --    résultat : l'état du bouton
 ------------------------------------------------------------------------------
 procedure ChangerTexte (              --    Permet de modifier l'attribut
   F : in out TR_Fenetre;              --    texte d'un élément de F
   NomElement : in String;             --    le nom de l'élément à modifier
   NouveauTexte : in String );         --    la nouvelle valeur de texte
 ------------------------------------------------------------------------------
 procedure ChangerContenu (            --    Permet de changer le contenu d'un champ
   F : in out TR_Fenetre;              --    de saisie ou d'une zone texte avec ascenseur
   NomElement : in String;             --    le nom de l'élément
   NouveauContenu : in String );       --    la nouvelle valeur de contenu
 ------------------------------------------------------------------------------
 procedure ChangerCouleurTexte (       --    Permet de changer la couleur
   F : in out TR_Fenetre;              --    du texte d'un element de F
   NomElement : in String;             --    le nom de l'élément à modifier
   NouvelleCouleur : in T_Couleur ) ;  --    la nouvelle couleur !
 ------------------------------------------------------------------------------
 procedure ChangerCouleurFond (        --    Permet de changer la couleur de fond
   F : in out TR_Fenetre;              --    d'un élément de F
   NomElement : in String;             --    le nom de l'élément à modifier
   NouvelleCouleur : in T_Couleur );   --    la nouvelle couleur !
 ------------------------------------------------------------------------------
 procedure ChangerStyleTexte (         --    Permet de changer le style du texte
   F : in out TR_Fenetre;              --    d'un élément de F
   NomElement : in String;             --    le nom de l'élément à modifier
   NouveauStyle : in FL_TEXT_STYLE );  --    le nouveau style !
 ------------------------------------------------------------------------------
 procedure ChangerTailleTexte (        --    Permet de changer la taille du texte
   F : in out TR_Fenetre;              --    d'un élément de F
   NomElement : in String;             --    le nom de l'élément à modifier
   Taille : in X11.Int );              --    La nouvelle taille !
 ------------------------------------------------------------------------------
 procedure ChangerEtatBouton (         --    Permet d'activer ou de désactiver un bouton
   F : in out TR_Fenetre;              --    ou une boîte à cocher de F
   NomElement : in String;             --    Le nom de l'élément à modifier
   Etat : in T_EtatBouton );           --    valeur : marche (activé) ou arret
 ------------------------------------------------------------------------------
 procedure ChangerImageBouton (        --    Change l'image d'un bouton image
   F : in out TR_Fenetre;              --    de F
   NomElement,                         --    le nom de l'élément à modifier
   NomImage : in String );             --    le nom du fichier contenant l'image
 ------------------------------------------------------------------------------
 procedure MontrerElem (F: in out TR_Fenetre; NomElement : in String );
 -- Permet de montrer un élément jusque-là caché
 ------------------------------------------------------------------------------
 procedure CacherElem (F: in out TR_Fenetre; NomElement : in String );
 -- Permet de cacher un élément jusque-là visible
 ------------------------------------------------------------------------------

end p_fenbase;
```

Annexe II – Marketing.adb

```
with p_fenbase ; use p_fenbase ;
with Forms ; use Forms;
with p_esiut; use p_esiut;
with ada.calendar; use ada.calendar;

procedure marketing is
    FSaisieNom, FJeu, FResultat : TR_Fenetre; -- l'application comporte 3 fenêtres
    Compteur : Natural;
    Touche : Character;
    I, J : Natural;
    Nombouton     : String(1..2);
    HeureDeb,HeureFin : Time;
    NewLine : constant Character := Character'Val (10); -- retour chariot
begin
    -- initialisation de l'interface graphique (à ne faire qu'une fois en début de programme)
    InitialiserFenetres;

    -- création de la fenêtre de saisie du nom du "gogo"
    FSaisieNom:=DebutFenetre("Nom du Joueur",400,70);
        AjouterChamp(FSaisieNom,"ChampNom","Votre Nom","quidam",100,10,280,30);
        AjouterBouton(FSaisieNom,"BoutonValider","valider",100,50,70,30);
        AjouterBouton(FSaisieNom,"BoutonAnnuler","annuler",180,50,70,30);
    FinFenetre(FSaisieNom);

     -- création de la fenêtre pour "jouer"
    FJeu:=DebutFenetre("CHANCE",300,400);
        AjouterTexte(FJeu,"message1","TAPE SUR 3 TOUCHES AU HASARD...",10,10,250,30);
        AjouterTexte(FJeu, "message2","Tente ta chance !",10,50,280,30);
        -- on ajoute une grille de 3x3 boutons affichant les valeurs de 1 à 9
        -- on donne à chaque bouton un nom d'élément qui représente sa position dans la grille
        -- le bouton de la ligne 2 colonne 2 s'appellera donc "23"
        for I in 1..3 loop
            for J in 1..3 loop
                nombouton := Integer'Image(I)(2..2) & Integer'Image(J)(2..2);
                -- le bouton "ij" affiche la valeur (i-1)*3+J
                AjouterBouton(FJeu,nombouton,integer'image((i-1)*3+J),(J-1)*60+40,(I-1)*60+90,60,60);
                -- modification de l'apparence du bouton (couleur de fond, taille et style du texte)
                ChangerCouleurFond(FJeu,nombouton,FL_DARKGOLD);
                ChangerTailleTexte(Fjeu,nombouton,FL_Large_Size);
                ChangerStyleTexte(Fjeu,nombouton,FL_Bold_Style);
            end loop;
        end loop;
        AjouterTexte(FJeu,"BarreDEtat","Aucune touche pressee",10,300,250,30);
        AjouterBouton(FJeu,"Fin","FIN",55,390,70,30);
        ChangerStyleTexte(FJeu, "Fin", FL_BOLD_Style);
        ChangerTailleTexte(Fjeu,"Fin",FL_medium_size);
        AjouterHorloge(Fjeu,"Clock","",150,320,100,100);
        AjouterBouton(FJeu,"BoutonAbandonner","abandon",55,350,70,30);
    FinFenetre(FJeu);

    -- création de la fenêtre d'affichage du "gain"
    Fresultat:=Debutfenetre("Votre gain ! ",300,150);
        AjouterTexteAscenseur(FResultat,"message","","",10,10,280,100);
        AjouterBouton(FResultat,"BoutonFin","The End",115,140,70,30);
    FinFenetre(FResultat);

    --------------------------------------------------------------------------------
    -- Affichage de la fenêtre de saisie et attente d'un bouton pressé
    MontrerFenetre(FSaisieNom);
    if AttendreBouton(FSaisieNom)/="BoutonAnnuler" then
        CacherFenetre(FSaisieNom);
        Compteur:=0; -- nombre de touches pressées
        -- modification du texte "message2" de la fenêtre de jeu :
        -- concaténation du contenu du champ de saisie du nom avec " tente ta chance !"
        ChangerTexte(Fjeu,"message2", ConsulterContenu(FSaisieNom,"ChampNom")& " tente ta chance !");
        -- Affichage de la fenêtre de jeu
        MontrerFenetre(FJeu);
        HeureDeb:=clock;
        -- Pour que le "gogo" soit obligé de cliquer sur 3 touches, on rend le bouton "Fin" inactif
        ChangerEtatbouton(FJeu, "Fin", Arret);
        loop -- BOUCLE jusqu'a la fin du jeu ou l'abandon
            declare
                Bouton : String := (Attendrebouton(Fjeu));
            begin
                if Bouton /= "BoutonAbandonner" and Bouton /= "Fin" then -- clic sur un bouton "ij"
                    if compteur < 3 then
                          Compteur:=Compteur+1; -- une nouvelle touche a été pressée
                          ChangerEtatBouton(FJeu, Bouton, Arret); -- bouton rendu inactif
                          ChangerCouleurFond(Fjeu, Bouton, FL_RED); -- et mis en rouge
                          ChangerTailleTexte(Fjeu,Bouton,FL_normal_size); -- pour que le texte rentre...
                          ChangerTexte(Fjeu, Bouton, "GAGNE"); -- et c'est l'arnaque !!!
                          -- on calcule les coordonnées (i,j) dans la grille du bouton pressé
                          -- grâce au nom du bouton...
                          I:=Character'Pos(Bouton(Bouton'First)) - Character'Pos('0');
                          J:=Character'Pos(Bouton(Bouton'Last)) - Character'Pos('0');
```

```
                        -- on calcule la "valeur" du bouton pressé et on la convertit en caractère
                        Touche:= Character'Val(((I-1)*3+J)+Character'Pos('0'));
                        -- on change l'objet texte "BarreDetat" en fonction de la touche pressée
                        ChangerTexte(Fjeu,"BarreDEtat","La touche " & Touche & " a ete pressee");
                        if compteur = 3 then -- C'est gagné !!!
                             -- on désactive toutes les touches (sans faire de détail !...)
                             for I in 1..3 loop
                                  for J in 1..3 loop
                                       nombouton := Integer'Image(I)(2..2) & Integer'Image(J)(2..2);
                                       ChangerEtatBouton(FJeu, nombouton, Arret);
                                  end loop;
                             end loop;
                             -- ainsi que le bouton "Abandonner"
                             ChangerEtatBouton(FJeu, "Abandonner", Arret);
                             HeureFin:= clock;
                             ChangerTexte(Fjeu,"message2","BRAVO !!! tu as gagne en"
                                       & natural'image(natural(Heurefin-HeureDeb))& "'");
                             ChangerStyleTexte(FJeu, "message2", FL_BOLD_Style);
                             ChangerTailleTexte(Fjeu,"message2",FL_medium_Size);
                             -- réactivation du bouton "Fin"
                             ChangerEtatBouton(FJeu,"Fin",Marche);
                        end if;
                   end if;
              elsif Bouton = "Fin" then
                   exit;
              else -- Bouton "Abandonner"
                   compteur := 0;
                   exit;
              end if;
         end;
    end loop;
    CacherFenetre(FJeu);
    if Compteur=0 then -- le joueur a abandonné : message "commercial"
         ChangerCouleurFond(FResultat,"fond",FL_BLACK);-- fond noir pour accabler
         ChangerContenu(FResultat,"message","Joueur " & ConsulterContenu(FSaisieNom,"ChampNom")
                        & NewLine & "qui ne tente rien n'a rien !!!");
    else -- le joueur a tenté sa chance, autre message "commercial"
         ChangerCouleurFond(FResultat,"fond",FL_RED); -- fond rouge pour encenser
         ChangerContenu(FResultat,"message","Joueur " & ConsulterContenu(FSaisieNom,"ChampNom")
                        & ", NOUS T'ATTENDONS ! " & NewLine & "RDV au magasin, "
                        & NewLine & "TU GAGNERAS ENCORE !!!");
    end if;
    -- Affichage de la fenêtre des gains
    Montrerfenetre(Fresultat);
loop
         exit when Attendrebouton(Fresultat) = "BoutonFin";
    end loop;
    Cacherfenetre(Fresultat);
end if;
end marketing;
```

Annexe III – Couleurs disponibles dans le paquetage Forms

FL_TOMATO, FL_INDIANRED, FL_SLATEBLUE, FL_COL1, FL_RIGHT_BCOL, FL_BOTTOM_BCOL

FL_TOP_BCOL, FL_LEFT_BCOL, FL_MCOL, FL_INACTIVE, FL_PALEGREEN, FL_ORCHID

FL_DARKCYAN, FL_DARKTOMATO, FL_WHEAT, FL_DARKORANGE, FL_DEEPPINK, FL_CHARTREUSE

FL_DARKVIOLET, FL_SPRINGGREEN, FL_DODGERBLUE, FL_DARKGOLD, FL_BLACK, FL_RED

FL_GREEN, FL_YELLOW, FL_BLUE, FL_MAGENTA, FL_CYAN, FL_WHITE

Annexe IV : Styles et tailles disponibles dans le paquetage Forms

TAILLE	***STYLE***
FL_TINY_SIZE (8 pt)	FL_NORMAL_STYLE – valeur par défaut
FL_SMALL_SIZE (10 pt)	FL_BOLD_STYLE (gras), FL_ITALIC_STYLE (italique)
FL_NORMAL_SIZE (12 pt)	FL_BOLDITALIC_STYLE (gras italique)
(valeur par défaut)	FL_FIXED_STYLE (tous les caractères occupent le même espace)
FL_MEDIUM_SIZE (14 pt)	FL_FIXEDBOLD_STYLE, FL_FIXEDITALIC_STYLE, FL_FIXEDBOLDITALIC_STYLE
FL_LARGE_SIZE (18 pt)	FL_TIMES_STYLE (police ⇔ Times ou Times New Roman déclinable en BOLD, ITALIC, BOLDITALIC)
FL_HUGE_SIZE (24 pt)	FL_SHADOW_STYLE (ombré) / EMBOSSED (relief) / ENGRAVED (gravé)
	Remarque : attention, certains choix de style peuvent rester sans effet (Pb. implémentation)