LIFAP1 – CC mi-parcours – Séquence 3

Contrôle Continu (Durée totale : 1h)
Jeudi 22 octobre 2020

NOM :

CORRIGE et BAREME

PRENOM :

.....

Numéro Etudiant :

.....

Recommandations : *Les documents, calculatrice, téléphone portable sont interdits. La qualité de l'écriture et de la présentation seront prises en compte dans la note finale. Vous veillerez à* ***respecter*** *les notations et les règles d'écriture des algorithmes vues en cours et en TD. Un soin tout particulier devra être apporté à l'écriture des entêtes des différents sous-programmes.*

Partie A – Langage C/C++

1. Ecrire en langage C/C++ une **fonction** booléenne `est_diviseur` qui renverra vrai si b divise a et faux sinon.
   Exemples : `est_diviseur(6,3)` renverra vrai et `est_diviseur(5,2)` renverra faux

2.5 points au total

```
bool est_diviseur (int a, int b)
{
   return (a % b == 0);
}
```

entête : 1 point

test 1 point

return 0.5 point

2. Ecrire en langage C/C++ une **procédure** `diviseurs_2_3_5` permettant de tester si un entier `n` strictement positif passé en paramètre est divisible par 2, par 3, et par 5. Cette procédure appellera le sous-programme précédent et "renverra" au programme principal 3 résultats booléens `div2`, `div3` et `div5`.

3.5 points au total

```
void diviseur_2_3_5 (int n, bool & div2, bool &div3, bool &div5)
{
   div2 = est_diviseur(n,2);
   div3 = est_diviseur(n,3);
   div5 = est_diviseur(n,5);
}
```

Entête : 2 points

Appels : 1.5 points

0.5 par appel

3. Ecrire en langage C/C++ le programme principal permettant :
    a. de demander un nombre à l'utilisateur et de le saisir,
    b. puis d'afficher les 3 valeurs obtenues en utilisant la procédure `diviseurs_2_3_5`.

4 points au total

```
int main(void)
{
   int val;
   bool d2,d3,d5;
   cout<<"Donnez un entier"<<endl;
   cin>>val;
   diviseur_2_3_5(val, d2,d3,d5);
   if (d2) cout<<val<<" est divisible par 2"<<endl;
      else cout<<val<<" n'est pas divisible par 2"<<endl;

   if (d3) cout<<val<<" est divisible par 3"<<endl;
      else cout<<val<<" n'est pas divisible par 3"<<endl;
   if (d5) cout<<val<<" est divisible par 5"<<endl;
      else cout<<val<<" n'est pas divisible par 5"<<endl;
   return 0;
}
```

déclaration des variables : 0.5

saisie de la valeur : 0.5 point

appel : 1 point

test + affichage 1 point pour le premier

0.5 pour les 2 suivants

Partie B – Algorithmique

1. Ecrire l'algorithme d'un sous-programme `saisie_valeur` qui demande à l'utilisateur une valeur **comprise entre 3 et 9 inclus** et la retourne. On recommencera la saisie tant que la valeur proposée ne respecte pas ces contraintes.

3 points

```
fonction saisie_valeur() : entier
préconditions : aucun
donnée : aucune
donnée / résultat : aucune
résultat : entier
description : saisit et retourne une valeur comprise entre 3 et 9
variable locale : val : entier

début
        faire
                afficher (« Donnez un entier paire strictement positif »)
                saisir (val)
        tant que ((val<3) ou (val>9))
        retourner val
fin
```

entête : 1 point

saisie : 0,5 point

boucle : 1 point

retour : 0.5 pt

2. Ecrire l'algorithme d'un sous-programme `dessine_triangle` qui permet d'afficher le motif ci-contre. La hauteur (une valeur comprise entre 3 et 9 inclus) sera passée en paramètre du sous-programme.
   On pourra utiliser `afficher(saut de ligne)` pour passer à la ligne suivante. Dans l'exemple ci-contre on aura : `dessine_triangle(7)`

```
1
22
333
4444
55555
666666
7777777
```

4 points

```
procédure dessine_triangle (h : entier)
préconditions : aucun
donnée : h                                          entête :  1 point
donnée / résultat : aucune
description : affiche le motif
variable locale : i, j : entier

début
        pour i allant de 1 à h par pas de 1 faire
                pour j allant de 1 à i par pas de 1 faire      double boucle : 2 points
                        afficher (i)                           affichage : 1 point
                fin pour
                afficher (saut de ligne)
        fin pour
fin
```

3. En utilisant les sous-programmes écrits en 1- et 2-, écrire l'algorithme du programme principal qui affiche un triangle avec une hauteur choisie par l'utilisateur.

```
Programme principal                                 3 points
Variables :
        hauteur : entier                            déclaration variable : 0.5 point
début
        hauteur ← saisie_valeur ()                  appel saisie_valeur : 1 point
        dessine_triangle (hauteur)                  appel dessin : 1 point
fin                                                 structure générale : 0.5 point
```