LIFAP1 – CC mi-parcours – Séquence 5

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 – Algorithmique

1. Ecrire l'algorithme d'un sous-programme `saisie_valeur` qui demande à l'utilisateur une valeur **strictement supérieure à 3 et impaire** 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                                              entête :  1 point
donnée / résultat : aucune
résultat : entier
description : saisit et retourne une valeur strictement supérieure à 3 et impaire
variable locale : val : entier

début
        faire
                afficher (« Donnez un entier paire strictement positif »)
                saisir (val)                                 saisie : 0,5 point
        tant que ((val<=3) ou (val modulo 2 = 0))            boucle : 1 point
        retourner val                                        retour : 0.5 pt
fin
```

2. Ecrire l'algorithme d'un sous-programme `dessine_triangle` qui permet d'afficher le motif ci-contre. La hauteur (une valeur supérieure à 3 et impaire) ainsi que les caractères seront passés en paramètres 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(5,'*','-')`

```
  *
 *-*
*---*
*-----*
*********
```

4.5 points

```
procédure dessine_triangle (h : entier, c1 : caractère, c2 : caractère)
préconditions : aucun
donnée : h, c1, c2                                          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 2*i-1 par pas de 1 faire        double boucle : 1.5 points
                        si (i =h) ou (j=1) ou (j=2*i-1) alors        test : 1.5 points
                                afficher (c1)
                        sinon afficher (c2)                          affichage : 0.5 point
                        fin si
                fin pour
                affiche (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 des caractéristiques choisies par l'utilisateur.

```
Programme principal                                         2.5 points
Variables :
        hauteur : entier
        car_pair, car_impair : caractères
début
        hauteur ← saisie_valeur ()                          appel saisie_valeur : 1 point
        afficher ("Donnez deux caractères")
        saisir (car_pair, car_impair)                       saisie des caractères : 0.5 point
        dessine_triangle (hauteur,car_impaire, car_pair)    appel dessin : 1 point
fin
```

Partie B – Langage C/C++

On souhaite écrire une version **simplifiée** du 421… L'ordinateur tirera aléatoirement 3 valeurs et recommencera l'opération tant que la combinaison est différente de 4 / 2 / 1.

1- Ecrire en langage C/C++ un sous-programme `combinaison_des` permettant de tirer aléatoirement 3 valeurs appelées `de1`, `de2`, et `de3` comprises entre 1 et 6 inclus. Ces valeurs ne seront pas affichées dans le sous-programme mais "renvoyées" au programme principal.

On utilisera la fonction C/C++ `rand( )` qui retourne une valeur aléatoire comprise entre 0 et une constante `RANDMAX` très grande.

2 points au total

```
void combinaison_des (int & de1, int &de2, int &de3)
{
   de1=rand()%6 +1;
   de2=rand()%6 +1;
   de3=rand()%6 +1;
}
```

entête 1.5 points

utilisation du rand : 0.5 point

2- Ecrire en langage C/C++ un sous-programme `tri_des` qui à partir des 3 entiers passés en paramètres les trie par **ordre décroissant**.
Exemple : si `de1=4  de2=5  de3=1`
on aura le résultat du tri suivant : `d1=5  de2=4  de3=1`

3 points au total

```
void ordonne_des (int & de1, int &de2, int &de3)
{int tampon;
     if (de1<de2)
    {
     tampon = de1;
     de1 = de2;
     de2 = tampon;
    }
 if (de1<de3)
    {
     tampon = de1;
     de1 = de3;
     de3 = tampon;
    }
 /* trier B et C */
 if (de2<de3)
    {
     tampon = de2;
     de2 = de3;
     de3 = tampon;
    }
}
```

entete 1.5 points

tri 1.5 points

3- Ecrire en langage C/C++ une **fonction booléenne** `verifie_combinaison` qui renverra vrai si la combinaison des 3 dés fournis en paramètres est 4, 2, et 1, faux sinon.

1.5 points au total

```
bool combinaison_gagnante (int de1, int de2, int de3)
{
   return ((de1==4)&&(de2==2)&&(de3==1));
}
```

entête 0.5 point

test + return : 1 point

4- Ecrire en langage C/C++ le programme principal permettant, en utilisant les 3 sous-programmes précédents, d'afficher le nombre de tirages nécessaires pour parvenir à la combinaison 4, 2, et 1.

4 points au total

```
int main (void)
{
   int v1,v2,v3;
   int nb_coups=0;
   srand(time(NULL));
   do
   {
      nb_coups++;
        combinaison_des(v1,v2,v3);
        ordonne_des(v1,v2,v3);
   }  while (!combinaison_gagnante(v1,v2,v3));
   cout<<"421 en "<<nb_coups<<" coups "<<endl;
        return 0;
}
```

déclaration + initialisation : 1 point

incrémentation 0.5 point

appels : 1 point

boucle 1 point

affichage 0.5 point