exercice 1 l'intersection
des intervalles

```
1  #include <iostream>
2  using namespace std;
3  int main ( ){
4     int A, B, C , D, start, end , tmp ;
5     //demande à l'utilisateur à saisir les
   entiers
6     cout << "entrez les valeurs de A,B,C et D
   : " << endl;
7     cin >> A>> B>> C>> D ;
8     // boucle pour tester si l'intervalle est
   correct ( bien ordonné)
9     if(A>B){
10       tmp=A;
11       A=B;
12       B=tmp;
13    }
14    if(C>D){
15       tmp=C;
16       C=D;
17       D=tmp;
18    }
19    // determiner le debut et fin de
   l'intervalle d'intersection pour connaitre
   leur limites
20    if(A>=C){
21       start=A;
22    }else{
23       start=C;
24       if (B<=D){
25          end=B;
26       }else
27       end=D;
28          // verification si on a l'inte      on
   ou non
```

(|) | [|] | = | -

exercice 1 (l'i...
/storage/emulate...

```
12      B=tmp;
13    }
14    if(C>D){
15      tmp=C;
16      C=D;
17      D=tmp;
18    }
19    // determiner le debut et fin de
l'intervalle d'intersection pour connaitre
leur limites
20    if(A>=C){
21      start=A;
22    }else{
23      start=C;
24      if (B<=D){
25        end=B;
26      }else
27      end=D;
28        // verification si on a l'intersection
ou non
29      if (start<=end){
30        cout << "l'intersection des
intervalles [" << A<<","<<B<<"]et [ " << C<<",
"<<D<<"] est l'intervalle :[ " << start <<"," <<
end <<"]"<< endl;
31      }
32      else
33      cout <<"les intervalles [ "<< A<<", "<<
B<< "] et [ " << C<< ", " << D<< "] ne
s'intersectent pas " << endl;
34  return 0;
35  }
36  }
```

(|) | [|] | = |

exercice 2:

la fonction de sin

```
1  #include <iostream>
2  using namespace std;
3  // la fonction qui calcule le factorielle
4  float Factorielle(int n) {
5      float fact = 1;
6      for (int i = 1; i <= n; i++) {
7          fact = fact * i;
8      }
9      return fact;
10 }
11 // fonction qui calcule puissance
12 double Puissance(int x, int n) {
13     double p = 1;
14     for (int i = 1; i <= n; i++) {
15         p = p * x;
16     }
17     return p;
18 }
19 // fonction qui calcule sinus en utilisant
   serie de Taylor
20 double CalculerSin(double x, int N) {
21     double S = 0;
22     for (int n = 0; n <= N; n++) {
23         double denominateur = Factorielle(2 *
    n + 1); // (2n+1)!
24     // on determine la parité du (-1)^n
25             int signe;
26         if (n % 2 == 0) {
27             signe = 1; // alors n est paire
28         } else {
29             signe = -1; // n est impair
30         }
31         double fct = signe * (Puissa          2 *
```

Tab | { | } | : | ;

exercice 2

```
26          if (n % 2 == 0) {
27              signe = 1; // alors n est paire
28          } else {
29              signe = -1; // n est impair
30          }
31          double fct = signe * (Puissance(x, 2 *
    n + 1) / denominateur); // (-1)^n ×[x^(2n+
    1) / (2n+1)! ]
32          S = S + fct; // la somme de la serie de
    n=0 jusqu'à N
33      }
34      return S;
35  }
36  // la fonction int main pour excécuter
    notre code
37  int main() {
38      double x;
39      int N;
40      // demande à l'utilisateur de saisir les
    valeurs
41      cout << "Entrez la valeur de x: ";
42      cin >> x;
43      cout << "Entrez le nombre de termes N:
    ";
44      cin >> N;
45
46      double result = CalculerSin(x, N);
47      cout << "Le sinus de " << x << " est
    approximativement: " << result << endl;
48
49      return 0;
50  }
51
```

Tab | { | } | : | ; | "

exercice3: de l'atelier 1

triangle isocèle des étoiles

```
1  #include<iostream>
2  using namespace std ;
3  int main(){
4    int L, i, j;
5    // demande à l'utilisateur de saisir le
   nombre de ligne
6    cout << "entrez le nombre de ligne du
   triangle: ";
7    cin >> L;
8   for (i=1; i<= L ; i++){ // boucle pour sauter
   à la ligne
9     for(j=1; j<= L-i ; j++) // boucle qui conter
   le nombre d'espace aprés *
10        cout << " ";
11     for (int k=1; k<=(2*i)-1;k++) // boucle
   pour les etoiles
12        cout << "*";
13    cout << endl;
14  }
15      return 0;
16 }
17
```

Tab | { | } | : | ;

exercice 4: de l'atelier 1

le jeu de dé


```
1 // exercice 4:
2 // Q1:
3 // Lorsque nous lançons un dé physique, nous obtenons un nombre parmi les valeurs possibles (1, 2, 3, 4, 5, ou 6). En informatique, on simule ce processus en générant un nombre aléatoire compris entre 1 et 6. Ce nombre représente le résultat du lancer de dé.L'idée est de générer un nombre aléatoire qui se trouve dans l'intervalle [1, 6]. Ce nombre correspondra au résultat du dé.
4 //En C++:
5 //#include <cstdlib>: Inclut la bibliothèque pour les fonctions mathématiques, notamment rand() qui génère des nombres aléatoires, mais par défaut.
6 //#include <ctime>: Inclut la bibliothèque pour manipuler le temps, utilisée ici pour initialiser le générateur de nombres aléatoires de manière différente à chaque exécution.
7
```

Tab | { | } | : | ; | "

```
#include <iostream>
#include <cstdlib>  // Pour rand() et srand()
#include <ctime>    // Pour time()
 using namespace std;

// Fonction qui simule un lancer de dé

int lancer_de() {
    return rand() % 6 + 1;  // Génère un nombre aléatoire entre 1 et 6 (Le modulo 6 (rand() % 6) prend la valeur aléatoire générée et la ramène à un intervalle de 0 à 5.En ajoutant 1, vous obtenez un nombre dans l'intervalle de 1 à 6, ce qui correspond aux faces d'un dé).
}

// Q2: Fonction throwDice permettant de lancer 1 ou 2 dés

int throwDice(int Lance) {
    int de1 = lancer_de();  // Lancer du premier dé
    if (Lance == 1) {
        cout << "Vous avez lancé un dé: " << de1 << endl;
        return de1;
    } else if (Lance == 2) {
        int de2 = lancer_de();  // Lancer du second dé
        cout << "Vous avez lancé de : " << de1 << " et " << de2 << endl;
```

Tab | { | } | : | ; | "

```
18        return de1;
19    } else if (Lance == 2) {
20        int de2 = lancer_de();  // Lancer du
    second dé
21        cout << "Vous avez lancé deux dés: "
    << de1 << " et " << de2 << endl;
22        if (de1 == de2) {
23            cout << "Dés égaux ! Vous perdez
    la somme des deux dés." << endl;
24            return -(de1 + de2);  // Score
    diminué de la resultat si les dés sont
    égaux
25        } else {
26            return de1 + de2;  // Score
    augmenté par la somme des dés
27        }
28    }
29    return 0;
30 }
31
32
33 // Q3: Fonction playerTurn correspondant
    au tour d'un joueur
34
35 int playerTurn(int joueur) {
36    int Lance;
37    // Boucle pour s'assurer que le joueur
    choisir  1 ou 2 lance
38    do {
39        cout << "Joueur " << joueur << ",
    voulez-vous lancer 1 ou 2 dés ? Entrez 1
    ou 2: ";
40        cin >> Lance;
```

Tab | { | } | : | ;

```
32
33  // Q3: Fonction playerTurn correspondant
    au tour d'un joueur
34
35  int playerTurn(int joueur) {
36      int Lance;
37      // Boucle pour s'assurer que le joueur
    choisir  1 ou 2 lance
38      do {
39          cout << "Joueur " << joueur << ",
    voulez-vous lancer 1 ou 2 dés ? Entrez 1
    ou 2: ";
40          cin >> Lance;
41      } while (Lance != 1 && Lance != 2);
42
43      // Appel de la fonction throwDice pour
    simuler le lancer
44      return throwDice(Lance);
45  }
46
47
48  // Q4: Programme principal (main) pour
    gérer le déroulement complet du jeu
49
50  int main() {
51      srand(time(0));  // Initialiser le
    générateur de nombres aléatoires
52
53      int score1 = 0;  // Score du Joueur 1
54      int score2 = 0;  // Score du Joueur 2
55      const int scoreMax = 30;  // Sc
    nécessaire pour gagner
```

Tab | { | } | : | ; | "

```
46
47
48  // Q4: Programme principal (main) pour
    gérer le déroulement complet du jeu
49
50  int main() {
51      srand(time(0));  // Initialiser le
    générateur de nombres aléatoires
52
53      int score1 = 0;  // Score du Joueur 1
54      int score2 = 0;  // Score du Joueur 2
55      const int scoreMax = 30;  // Score
    nécessaire pour gagner
56
57      // Boucle principale du jeu, continue
    tant qu'aucun joueur n'a atteint 30 points
58      while (score1 < scoreMax && score2 <
    scoreMax) {
59          // tour de joueur 1
60          int resultat1 = playerTurn(1);  // Appel
    de la fonction playerTurn pour Joueur 1
61          score1 += resultat1;
62          cout << "Score du Joueur 1: " <<
    score1 << endl;
63          // Vérification si le Joueur 1 a atteint
    ou dépassé 30 points
64          if (score1 >= scoreMax) break;
65          // Tour du Joueur 2
66          int resultat2 = playerTurn(2);  // Appel
    de la fonction playerTurn pour Joueur 2
67          score2 += resultat2;
68          cout << "Score du Joueur 2:
    score2 << endl;
```

Tab | { | } | : | ; |

```
62        cout << "Score du Joueur 1: " << score1 << endl;
63        // Vérification si le Joueur 1 a atteint ou dépassé 30 points
64        if (score1 >= scoreMax) break;
65        // Tour du Joueur 2
66        int resultat2 = playerTurn(2);  // Appel de la fonction playerTurn pour Joueur 2
67        score2 += resultat2;
68        cout << "Score du Joueur 2: " << score2 << endl;
69        // Vérification si le Joueur 2 a atteint ou dépassé 30 points
70        if (score2 >= scoreMax) break;
71    }
72    // Affichage du résultat final
73    cout << "\n ## Résultat final ##" << endl;
74    cout << "Score final du Joueur 1: " << score1 << endl;
75    cout << "Score final du Joueur 2: " << score2 << endl;
76    // Détermination du gagnant ou match nul
77    if (score1 >= scoreMax && score2 >= scoreMax) {
78        cout << "C'est un match nul !" <<endl;
79    } else if (score1 >= scoreMax) {
80        cout << "Le Joueur 1 a gagné !" <<endl;
81    } else if (score2 >= scoreMax) {
82     cout << "Le Joueur 2 a gagné !" <<endl;
83    }
84    return 0;
85 }
```

Tab | { | } | : | ; | "

exercice 2 de l'atelier 1

```
1  // le programme s'affiche
2
3  A: n = 10 p = 10 q = 10 r = 1 (vraie10==10)
4  B: n = 10 p = 5 q = 5
5  D: n = 11 p = 5 q = 10
6  D: n = 6 p = 3 q = 1( vraie n=6 et p=3)
7
```

Tab | { | } | : | ; | "