Jeu de l'oie [it11] - Exercice

Karine Zampieri, Stéphane Rivière, Béatrice Amerein-Soltner

Unisciel algoprog Université Haute-Alsace

Version 17 mai 2018

Table des matières

C++ - Jeu de l'oie (Solution)

Mots-Clés Schéma itératif ■
Requis Structures de base, Structures conditionnelles, Algorithmes paramétrés, Structures répétitives ■
Difficulté ●●○ (45 min à 1 h) ■

Objectif
Cet exercice simule les mouvements du pion d'un seul joueur au jeu de l'oie.

1 Jeu de l'oie / pgjeudeloie

1.1 Présentation du Jeu de l'Oie

Le principe de ce jeu d'enfants est de déplacer un pion (jet de deux dés) par joueur sur une spirale de cases numérotées de 1 à 63 dont certaines d'entre elles jouent un rôle particulier. (Jeu de l'Oie (wikipedia))

Règle 1
Celui qui du premier coup, jette 6 et 3, se place au numéro 26, et celui qui jette 4 et 5 va se placer au numéro 53.

Règle 2
Le joueur qui arrive sur une oie (numéro de case divisible par 9) avance d'autant de points qu'il vient d'amener.

Règle 3
Celui qui s'arrête sur le pont (n°6) va se placer au numéro 12. Le joueur qui arrive à l'Hôtellerie (n°19) laisse passer deux tours sans jouer. Celui qui tombe dans le puits (n°31) reste jusqu'à ce qu'il en soit délivré par un autre qui prend sa place. Celui qui arrive au labyrinthe (n°42) retourne au numéro 30. Le joueur qui arrive dans la prison (n°52) attend sa délivrance. Celui qui va trouver la mort (n°58) retourne en 1.

Règle 4
Celui qui est rejoint par un autre joueur va se mettre d'où ce dernier est venu.

Règle 5
Le joueur qui dépasse le numéro 63 retourne d'autant de points qu'il a jetés en trop, et s'il vient sur une oie, il rétrograde encore d'autant de points qu'il a jeté.

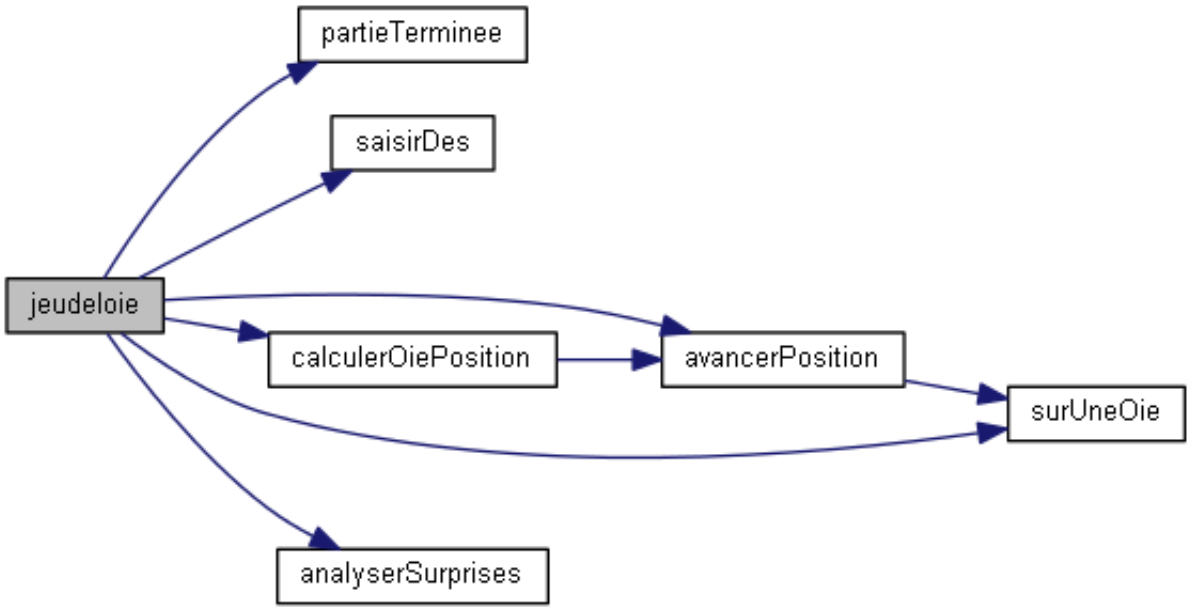
Règle 6
Celui qui atteint exactement le numéro 63 gagne la partie.

1.2 Pseudo-code du jeu

L'exercice simule les mouvements du pion d'un seul joueur au jeu de l'oie : il demande les entiers correspondant au jet des deux dés et calcule la nouvelle position du pion. Il affiche un message après chaque calcul de position pour signaler qu'il peut y avoir déjà un autre joueur sur la case qu'il vient atteindre. Voici un exemple du résultat attendu :

```
Vous etes en 0
Regardez s'il y a un pion
Les valeurs de vos deux dés? 4 5
Premier coup 4 + 5 -- Allez en 53
Calcul 4 5 ==> 53
Case banale
Vous etes en 53
Regardez s'il y a un pion
Les valeurs de vos deux dés? 6 4
Bravo! Vous avez gagné sauf si un autre joueur est déjà là
```

En vous aidant du graphe des appels et de l'exemple, écrivez le pseudo-code du mouvement du pion d'un joueur au jeu de l'oie.



Solution simple

```
position <- 0 // position effective
TantQue (non partieTerminee(position)) Faire
| saisirDes(de1,de2)
| avancerPosition(position,de1,de2)
| Si (non partieTerminee(position)) Alors
| | Si (surUneOie(position)) Alors
| | | calculerOiePosition(position,de1,de2)
| | FinSi
```

```
| | analyserSurprises(position)
| Finsi
FinTantQue
afficher résultat
```

1.3 Opération partieTerminee

Ce problème réalise :

- Règle 6 : Celui qui atteint exactement le numéro 63 gagne la partie.

Définissez la constante entière `DERNIERECASE=63` (numéro de la dernière case).

Écrivez une opération `partieTerminee(position)` qui teste et renvoie `Vrai` si la `position` (entier) désigne la dernière case.

Validez votre opération avec la solution.

Solution C++ @[pgjeudeloie.cpp]

```
/**
  Dernière case du jeu
*/

const int DERNIERECASE = 63;

/**
  Prédicat de fin de partie
  @param[in] position - position du pion
  @return Vrai si la partie est terminée, Faux sinon
*/

bool partieTerminee(int position)
{
  return (position == DERNIERECASE);
}
```

1.4 Opération avancerPosition

Ce problème réalise :

- Règle 2 : Le joueur qui arrive sur une oie (numéro de case divisible par 9) avance d'autant de points qu'il vient d'amener.
- Règle 5 : Le joueur qui dépasse le numéro 63 retourne d'autant de points qu'il a jetés en trop, et s'il vient sur une oie, il rétrograde encore d'autant de points qu'il a jeté.

Écrivez une opération `surUneOie(position)` qui teste et renvoie `Vrai` si la `position` (entier) est sur une oie.

Solution simple
On teste si la position est divisible par 9.

Écrivez une opération `avancerPosition(position,de1,de2)` qui, étant donnée la `position` (entier) actuelle, avance la position issue du jet de deux dés `de1` (entier) et `de2` (entier).

Solution simple
On calcule la position théorique dans `calculpos` (entier) puis on teste si `calculpos` est au-delà de la dernière case, auquel cas on applique la règle 5.

Validez vos opérations avec la solution.

Solution C++ @[pgjeudeloie.cpp]

```
/**
  Prédicat d'un pion sur une oie
  @param[in] position - position du pion
  @return Vrai si position est sur une oie, Faux sinon
*/

bool surUneOie(int position)
{
  return (position % 9 == 0);
}
```

```
/**
  Avance une position
  @param[in,out] position - position du pion
  @param[in] de1 - jet du dé 1
  @param[in] de2 - jet du dé 2
*/

void avancerPosition(int& position, int de1, int de2)
{
  int calculpos = position + (de1 + de2);
  if (calculpos > DERNIERECASE)
  {
    position = DERNIERECASE - (calculpos - DERNIERECASE);
    if (surUneOie(position))
    {
      position -= (de1 + de2);
    }
  }
  else
```

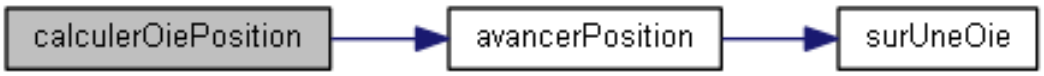
```
  {
    position = calculpos;
  }
}
```

1.5 Opération calculerOiePosition

Ce problème réalise :

- Règle 1 : Celui qui du premier coup, jette 6 et 3, se place au numéro 26, et celui qui jette 4 et 5 va se placer au numéro 53.
- Règle 2 : Le joueur qui arrive sur une oie (numéro de case divisible par 9) avance d'autant de points qu'il vient d'amener.

Écrivez une opération `calculerOiePosition(position,de1,de2)` qui, étant donnée la `position` (entier) actuelle sur une oie, calcule la nouvelle position issue du jet de deux dés `de1` (entier) et `de2` (entier).



Solution simple

On teste le cas du premier jet (règle 1). Sinon on avance la `position` (règle 2).

Validez votre opération avec la solution.

Solution C++ @[pgjeudeloie.cpp]

```
/**
  Calcule la position sur une oie
  @param[in,out] position - position du pion
  @param[in] de1 - jet du de 1
  @param[in] de2 - jet du de 2
*/

void calculerOiePosition(int& position, int de1, int de2)
{
  if (position == 9)
  {
    if (de1 == 6 || de2 == 6)
    {
      cout<<"Premier coup 6 + 3 -- Allez en 26"<<endl;
      position = 26;
    }
    else
    {
      cout<<"Premier coup 4 + 5 -- Allez en 53"<<endl;
      position = 53;
    }
```

```
  }
  else
  {
    cout<<"Sur une oie -- Avancez de "<<(de1 + de2)<<endl;
    avancerPosition(position,de1,de2);
  }
}
```

1.6 Opération analyserSurprises

Ce problème réalise la règle 3 :

1. Celui qui s'arrête sur le pont (n°6) va se placer au numéro 12.
2. Le joueur qui arrive à l'Hôtellerie (n°19) laisse passer deux tours sans jouer.
3. Celui qui tombe dans le puits (n°31) reste jusqu'à ce qu'il en soit délivré par un autre qui prend sa place.
4. Celui qui arrive au labyrinthe (n°42) retourne au numéro 30.
5. Le joueur qui arrive dans la prison (n°52) attend sa délivrance.
6. Celui qui va trouver la mort (n°58) retourne en 1.

Écrivez une opération `analyserSurprises(position)` qui, étant donnée la `position` (entier) actuelle, analyse les surprises de l'éventuelle nouvelle position.

Solution simple

On teste la `position` par rapport aux numéros n°6 (pont), n°19 (Hôtellerie), n°31 (puit), n°42 (labyrinthe), n°52 (prison) et n°58 (mort) (structure `Si` en cascade par exemple).

Validez votre opération avec la solution.

Solution C++ @[pgjeudeloie.cpp]

```
/**
  Analyse les surprises
  @param[in,out] position - position du pion
*/

void analyserSurprises(int& position)
{
  if (position == 6)
  {
    cout<<"Pont -- allez en 12"<<endl;
    position = 12;
  }
  else if (position == 19)
  {
    cout<<"Hotellerie -- Passez 2 tours"<<endl;
  }
  else if (position == 31)
```

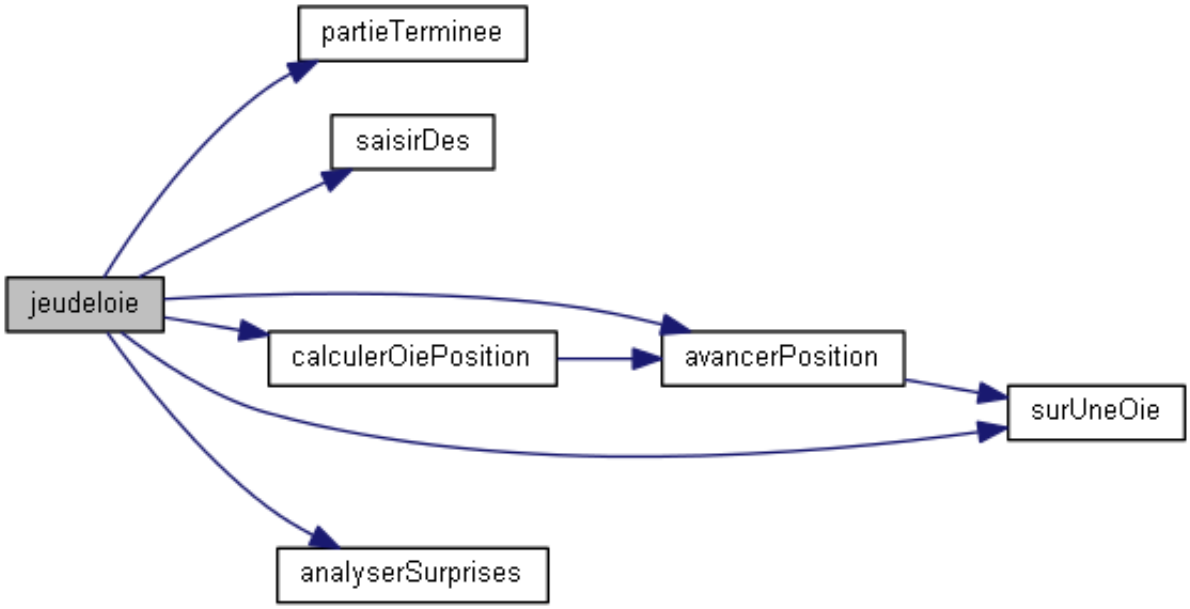
```
  {
    cout<<"Puis -- Patience!"<<endl;
  }
  else if (position == 42)
  {
    cout<<"Labyrinthe -- allez en 30"<<endl;
    position = 30;
  }
  else if (position == 52)
  {
    cout<<"Prison -- Patience!"<<endl;
  }
  else if (position == 58)
  {
    cout<<"Mort -- allez en 1"<<endl;
    position = 1;
  }
  else
  {
    cout<<"Case banale"<<endl;
  }
}
```

1.7 Programme principal

Écrivez une opération `saisirDe()` qui effectue la saisie validée d'un jet de dé à six faces. Affichez l'invite :

```
Valeur du de?
```

Écrivez une opération `jeudeloie` qui demande les entiers correspondant au jet des deux dés (dans `de1` (entier) et dans `de2` (entier) par exemple), puis calcule la nouvelle position du pion. Il affiche un message après chaque calcul de position pour signaler qu'il peut y avoir déjà un autre joueur sur la case qu'il vient atteindre.



Testez. Exemples d'exécution :

```
Vous etes en 0
Regardez s'il y a un pion
Les valeurs de vos deux dés? 4 5
Premier coup 4 + 5 -- Allez en 53
Calcul 4 5 ==> 53
Case banale
Vous etes en 53
Regardez s'il y a un pion
Les valeurs de vos deux dés? 6 4
Bravo! Vous avez gagné sauf si un autre joueur est déjà là
```

```
Vous etes en 0
Regardez s'il y a un pion
Les valeurs de vos deux dés? 3 3
Calcul 3 3 ==> 6
Pont -- allez en 12
Vous etes en 12
Regardez s'il y a un pion
Les valeurs de vos deux dés? 6 1
Calcul 6 1 ==> 19
Hotellerie -- Passez 2 tours
Vous etes en 19
Regardez s'il y a un pion
Les valeurs de vos deux dés? 6 6
Calcul 6 6 ==> 31
Puis -- Patience!
Vous etes en 31
Regardez s'il y a un pion
Les valeurs de vos deux dés? 5 6
```

```
Calcul 5 6 ==> 42
Labyrinthe -- allez en 30
Vous etes en 30
Regardez s'il y a un pion
Les valeurs de vos deux dés? 2 3
Calcul 2 3 ==> 35
Case banale
Vous etes en 35
Regardez s'il y a un pion
Les valeurs de vos deux dés? 5 5
Sur une oie -- Avancez de 10
Calcul 5 5 ==> 55
Case banale
Vous etes en 55
Regardez s'il y a un pion
Les valeurs de vos deux dés? 4 4
Bravo! Vous avez gagné sauf si un autre joueur est déjà là
```

Validez votre programme avec la solution.

Solution C++ @[pgjeudeloie.cpp]

```
/**
  Saisie validée d'un dé (6 faces)
  @return jet du de
*/

int saisirDe()
{
  int de = 0;
  while (!(1 <= de && de <= 6))
  {
    cout<<"Valeur du de? ";
    cin>>de;
  }
  return de;
}
```

```
/**
  Lance la simulation du jeu
*/

void jeudeloie()
{
  int position = 0;
  while (!partieTerminee(position))
  {
    cout<<"Vous etes en "<<position<<endl;
    cout<<"Regardez s'il y a un pion"<<endl;
    int de1 = saisirDe();
    int de2 = saisirDe();
    avancerPosition(position,de1,de2);
```

```
    if (!partieTerminee(position))
    {
      if (surUneOie(position))
      {
        calculerOiePosition(position,de1,de2);
      }
      cout<<"Calcul "<<de1<<" "<<de2<<" ==> "<<position<<endl;
      analyserSurprises(position);
    }
  }
  cout<<"Bravo! Vous avez gagne sauf si un autre joueur est deja la"<<endl;
}
```

2 Références générales

Comprend [Grogono :c5] ■