Mini Projet C++

Pierre & Hugo & Hugo Kueny

Groupe J

09/10/2017

IUT-ORSAY

Mini Projet C++

Jeu de Nim

Table des matières

Introduction

L'objectif de ce projet est de créer une variante du jeu de Nim en c++, où les joueurs tirent à tour de rôle autant de bâtonnets qu'ils souhaitent dans 3 tas distincts. Le joueur qui retire le dernier bâtonnet gagne la partie.

Hypothèses

- Nos hypothèses sont les suivantes :
  1. L'intelligence artificielle n'a pas le droit de piocher un tas entier.
  2. Le joueur n°1 ou le perdant du match précédent rejoue le premier.
  3. Chaque joueur doit retirer au moins un bâtonnet par tour.

- Choix de programmation :

Nous avons décidé de commencer par développer la partie multi-joueurs étape par étape. Une évolution du projet permet de jouer tout seul contre une intelligence artificielle, qui progresse au fil du jeu.

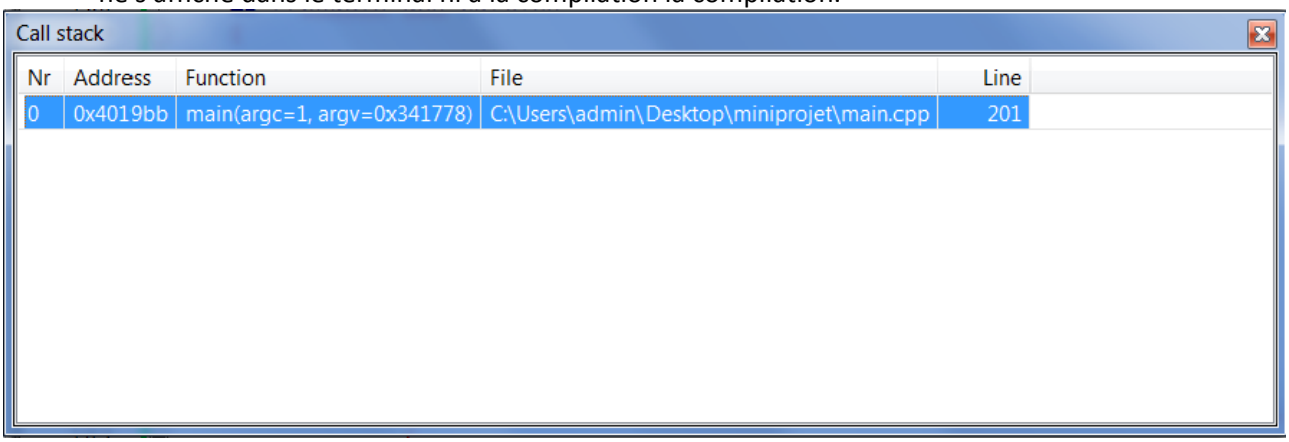
Utilisation de la librairie C pour l'accès au fichiers.

Difficultés Rencontrées

La plus grande difficulté de ce projet a été de vérifier tous les cas possibles afin de corriger les dernières erreurs.

Voici deux exemples concrets des dernières erreurs rencontrées :

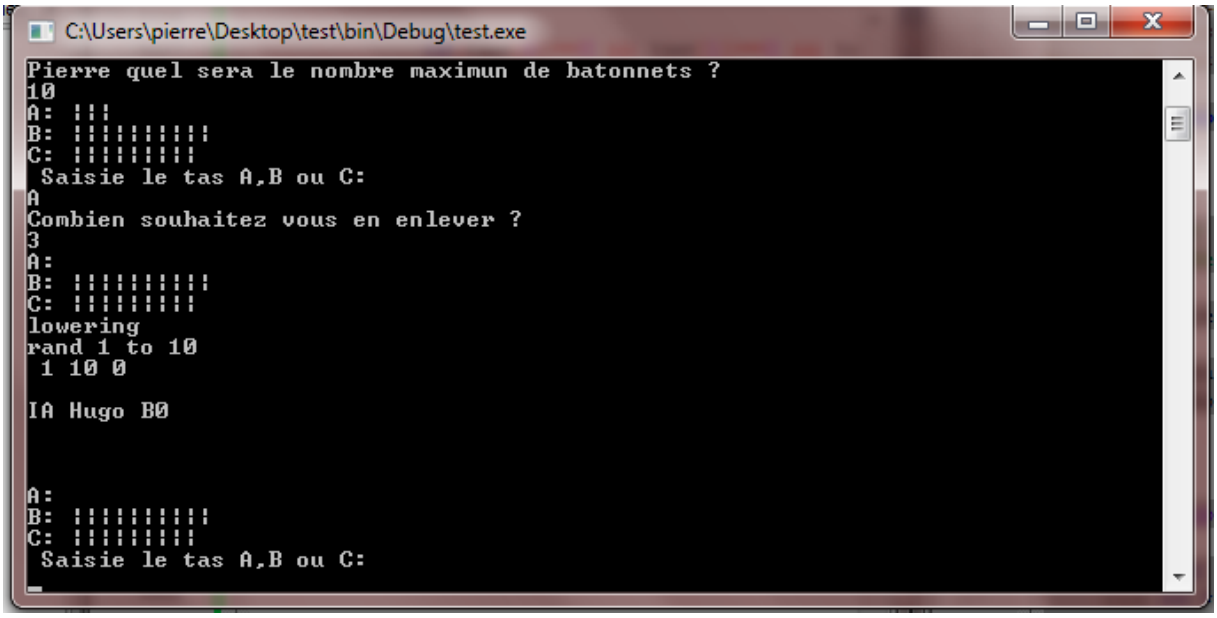
-Le programme cesse de fonctionner dans le terminal quand on joue contre l'intelligence artificielle au niveau 1. Le phénomène se produit alors qu'aucune erreur ne s'affiche dans le terminal ni à la compilation la compilation.



Pour localiser le bug, des traces ont été ajoutées aux étapes-clés pour en détecter la cause. Malgré la reproduction précise des conditions initiales, il a été impossible de définir la raison du plantage avant qu'il ne se produise.

Il a fallu finalement activer l'option debug de CodeBlocks pour détecter une division par zéro, valeur prise par une de nos variables en cours d'exécution.

Un autre problème est apparu au cours des tests :
L'intelligence artificielles ne respectait pas la règle n°3, à savoir l'obligation de retirer au moins un bâtonnet, lorsque c'était son tour de jouer.



Cette erreur provenait de la limite des bâtonnets qui était réglée à 0 au lieu du max.

Tableau récapitulatif des tests

Objectif	Nb de tests	Valeurs testées	IA actif	Nombre de bâtonnets	Résultat
Tester qu'il y'a pas de soucis sur le prénom des joueurs	5	Pierre, Hugo, 23, !, Kevin.	0	0	Réussite
Changement Joueur a la fin d'un tour	2	Pierre / Hugo Pierre/IA Hugo	0 1	5 5	Réussite Réussite
Changement du nom du perdant à la fin d'une partie	2	Pierre / Hugo Pierre/IA Hugo	0 1	5 5	Réussite Réussite
Test victoire si tous les tas sont vides	6	Finir Vider tas A, B ou C	0/1	10	Réussite
Test victoire si un tas est vide	6	Pioche tas A, B ou C Vide	0/1	5	Réussite
Test nombre bâtons max	14	1,2,3,5,10,15,20	0/1	1,2,3,5,10,15,20	Réussite
Test niveaux IA	12	1,2,3,4,5,6,7	1	3,5	Réussite
Entrer des quantités non-numériques sur le tas choisi	30	D, E, 5, !, P	0/1	5	Réussite
Entrer des quantités non-numériques sur le nombre de bâtonnets à enlever	18	E, e, ! ,	0/1	5	Réussite
Test de mauvaises valeurs dans un tas	48	-1,0,6,10	0/1	3,5	Réussite
Test sauvegarder données Joueurs	2	Pierre, Hugo Pieeeerre, Kevin	0/1	3	Réussite
Test de récupération des données	2	Pierre, Hugo Pieeeerre, Kevin	0/1	5	Réussite
Test de perte d'un niveau en cas de defaite contre l'IA	7	Lvl 1,2,3,4,5,6,7	1	5	Réussite

Objectif	Nb de tests	Valeurs testées	IA actif	Nombre de bâtonnets	Résultat
Nombre de bâtons maximum pouvant être enlever par tour	14	0,1,2,3,5,6,10	0/1	5	Réussite
Test de gain de point en mode multijoueur	2	Pierre Victoire / Hugo Victoire	0	5	Réussite
Test de perte de point en mode multijoueur	2	Hugo defaite / Pierre Defaite	0	5	Réussite

Extension

Après concertation nous avons donc décidé de rajouter quelques options supplémentaires :

- Paramétrer le nombre maximum de bâtonnets que l'on peut enlever en une seule fois.
- Faire progresser l'Intelligence Artificielle avec le joueur (Niveau qui progression de 1 à 7 avec des stratégies différentes)
- Ajouter une sauvegarde de la progression du joueur quand il joue contre l'intelligence artificielle.
- Le changement des Joueurs lors d'une nouvelle partie.
- Un système de « Nim Point » dans le mode multijoueur.

Conclusion

Avec 165 tests, ce projet semble abouti et complet malgré les difficultés rencontrées durant la phase de conception.

Ce projet nous a permis de consolider nos compétences en c++, mais aussi à améliorer notre capacité à travailler en binôme, tant dans la répartition des tâches que dans la gestion du temps imparti.

Code Source

Code Source

```
#include <iostream>
#include <limits.h>
#include <limits>
#include <string.h>
#include <cstdlib>
#include <time.h>
#include <cstdio>
#include <string>
#include <unistd.h>
#include <assert.h>

#ifdef _WIN32

#include <windows.h>
#define sleep(a) Sleep(a)

#endif // _WIN32


using namespace std;



/* nettoyer cin
 * But :  Nettoyer 'cin' pour prévenir les boucles infinies en cas d'entrée de mauvais type
 */
void nettoyer_cin()
{
    cin.clear(); // on retire l'état d'erreur
    cin.ignore(numeric_limits<streamsize>::max(), '\n'); // on oublie la denière ligne
}

/* demander
 *  Entrée: result=pointeur vers la variable a remplir  question=texte qui posera la question a l'utilisateur
 *  Sortie: etat d'erreur, vrai=saisie incompatible avec la variable passee ("bonjours" dans un int)
 *  But: poser une question et recuperer une reponse
 */
template<typename V> bool demander(V * result, string question)
{
    cout << question << endl << " ";
    cin >> *result;
    return cin.fail();
}
```

```
	/* demander_interval
	 *  Entrée: result=pointeur vers la variable a remplir
	 *		min=valeur minimale acceptee
	 *		max=valeur maximale acceptee
	 *		question=texte qui posera la question a l'utilisateur
	 *  But: poser une question et recuperer une reponse valide, comprise entre min et max
	 */
	template<typename V>
	void demander_interval(V * result, int min, int max, string question)
	{
		while(demander(result, question) || (min != -1 && *result < min) ||  (max != -1 && *result > max))
		{
			cout << "Reponse erronee" << endl;
			nettoyer_cin();
		}
		nettoyer_cin();
	}

	/* demander_lettre
	 *  Entrée: min=valeur minimale acceptee
	 *		max=valeur maximale acceptee
	 *		question=texte qui posera la question a l'utilisateur
	 *  Sortie: une lettre
	 *  But: poser une question et recuperer une lettre valide, comprise entre min et max
	 */
	char demander_lettre(char min, char max, string question)
	{
		char result;
		demander_interval(&result,min,max,question);
		return result;
	}

	/* demander_lettre_sanscasse
	 *  Entrée: min=valeur minimale acceptee
	 *		max=valeur maximale acceptee
	 *		question=texte qui posera la question a l'utilisateur
	 *  Sortie: une lettre
	 *  But: poser une question et recuperer une lettre valide, comprise entre min et max, sans tenir compte de la casse
	 */
	char demander_lettre_sanscasse(char min, char max, string question)
	{
		char result;
		while(demander(&result, question) || !( ((result >= toupper(min)) && (result <= toupper(max))) ||  ((result >= tolower(min)) && (result <= tolower(max)))  )) // tant que la valeur entree n'est pas correcte
		{
			cout << "Reponse erronee" << endl;
			nettoyer_cin();
		}
```

```
    nettoyer_cin();
    return result;
}

/* demander_nombre
 *  Entrée: min=valeur minimale acceptee
 *        max=valeur maximale acceptee
 *        question=texte qui posera la question a l'utilisateur
 *  Sortie: un nombre
 *  But: poser une question et recuperer un nombre valide, compris entre min et max
 */
int demander_nombre(int min, int max, string question)
{
    int result;
    demander_interval(&result,min,max,question);
    return result;
}

/* demander_texte
 *  Entrée: question=texte qui posera la question a l'utilisateur
 *  Sortie: un string
 *  But: poser une question et recupere un string de texte
 */
string demander_texte(string question)
{
    string result;
    demander(&result,question);
    nettoyer_cin();
    return result;
}

/* demander_bool
 *  Entrée: question=texte qui posera la question a l'utilisateur
 *  Sortie: un booleen
 *  But: poser une question et recuperer une reponse booleene valide, accepte 'O' 'o' 'n'
'N' ou tout mot commenceant par une de ces lettres
 */
bool demander_bool(string question)
{
    char answer;
    while(demander(&answer, question) || ( answer!='O' && answer!='o' && answer!='N'
&& answer!='n' )) // tant que la valeur demandee n'est pas correcte
    {
        cout << "Reponse erronee" << endl;
    }

    nettoyer_cin();

    return (answer == 'o' || answer == 'O');
}

/* rand_juste
```

```
	 *  Entree: min= valeur minimum ou egal (MIN 0)     max= valeur max ou egal
	 *  Retourne: un nombre vraiment aleatoire entre min et max
	 *
	 */
	int rand_juste(int min, int max)
	{
	    if(max == 0) // si max == 0   =>   on ne peut que repondre 0
	    {
	        return 0;
	    }

	    return rand()%(max+1 -min) +(min); // on génère un nombre aléatoire entre min et
max
	}


	/*
	 * tas avec mini et diff
	 *  Entree: tas= le tableau de tas      mini= valeur minimum ou egal recherchee
	 *     different= numero de tas qui ne sera pas accepte
	 *  But: Retourne un tas vraiment aleatoire repondant au crtieres de valeur mini et de tas
refusé
	 *
	 */
	int tas_avec_mini_et_diff(int tas[3], int mini, int different)
	{
	    int id_list[3] = {-1,-1,-1}; // liste des tas correspondants
	    int id_cnt = 0; // nombre des tas correspondants

	    for(int i = 0 ; i < 3 ; i++) // pour tout les tas
	    {
	        if(tas[i] >= mini && i != different) // si le mini et le different sont respectés
	        {
	            id_list[id_cnt] = i; // on rajoute un tas
	            id_cnt++; // on augmente le compte
	        }
	    }

	    if(id_cnt == 0) // si il n'y a pas de correspondant
	    {
	        return -1; // on retourne -1
	    }

	    int choice = rand_juste(0,id_cnt-1); // on choisis un numero aleatoire dans la liste des
tas correspondants
	    return id_list[choice]; // on donne le tas correspondant au choix du dessus
	}



	/* tas le plus haut bas
	 * Entree: tas= le tableau de tas        est_bas= le tas recherché est le plus bas (vrai) ou
```

```
 *     haut (faux)
 * Sortie: l'id du tas le plus bas ou le plus haut selon le parametre.
 *
 */
int tas_le_plus_haut_bas(int tas[3], bool choix_bas)
{
    int id_bas; // numero du tas le plus bas
    int val_bas = 999999; //valeur du tas le plus bas
    int id_haut; // numero du tas le plus haut
    int val_haut = -1; // valeur du tas le plus haut

    for(int i = 0 ; i < 3 ; i++) // pour tout les tas
    {
            if(tas[i] < val_bas) // si le tas est plus bas que le plus bas
            {
                    val_bas = tas[i]; // on prend un nouveau plus bas
                    id_bas = i;
            }
            if(tas[i] > val_haut) // si le tas est plus haut que le plus haut
            {
                    val_haut = tas[i]; // on prend un nouveau plus haut
                    id_haut = i;
            }
    }

    if(choix_bas) // si on veut le plus bas
    {
            return id_bas; // on rend le tas le plus bas
    }
    else // si on veut le plus haut
    {
            return id_haut; // on rend le plus haut
    }

}

/* player_info
 *
 * classe contenant un nom d'utilisateur et ses niveaux
 */
class player_info
{
public:
    string name; // nom
    int level_IA; // niveau de l'IA
    int level_multi; // Nim points
};


player_info * players = NULL; // pointeur sur liste des informations des joueurs
int player_cnt = 0; // taille de la liste des informations des joueurs
```

```
#define SEPARATEUR 0x1B //charactere ASCII impossible a entrer dans cin
#define NOM_FICHIER "joueurs" // nom du fichier

/* load_file
 *  But: ouvre et lit le fichier de sauvegarde pour creer la liste des joueurs
 */
void load_file()
{
    FILE * file; // fichier qui represente le fichier
    file = fopen(NOM_FICHIER,"a"); // on ouvre le fichier pour le creer
    assert(file != NULL); // assert si le fichier n'est pas ouvert
    fclose(file); // fermer le fichier

    file = fopen(NOM_FICHIER,"r"); // on ouvre le fichier pour le lire
    assert(file != NULL); // assert si le fichier n'est pas ouvert


    string line = ""; //ligne actuelle
    bool done = false; //condition de fin
    do
    {
        char character; //lettre actuelle
        if(fread(&character, 1, 1, file) == 0) //lit une lettre, vrai quand la lecture est
imposible (=fin de fichier)
        {
            done = true; // fini
            continue;
        }

        line += character; //on ajoute le charactere actuel a la fin de la ligne actuelle

        if(character == '\n') // si la ligne est complete
        {
            players = (player_info *) realloc(players,
((player_cnt+1)*(sizeof(player_info ) )) ); //on rajoute un joueur
            assert(players != NULL); // si la memoire n'est pas allouee

            new (players+player_cnt) player_info(); // on initialise le nouveau
joueur

            players[player_cnt].name = ""; // on met le nom a 0

            int i;
            for(i = 0 ; i < line.length() ; i++) // sur toute la ligne
            {
                if(line[i] == SEPARATEUR) // si le charactere actuel est le
separateur
                {
                    break; // fin de la boucle
                }
                players[player_cnt].name += line[i]; // on ajoute le caractere
actuel au nom
```

```
                        }
                        i++;//on saute le separateur

                        if(players[player_cnt].name == "") // le nom est vide
                        {
                                cout << "JOUEUR COROMPU ! Rennomé \"sans_nom\"" <<
endl;
                                players[player_cnt].name = "sans_nom"; // on renomme en
sans_nom
                        }


                        char filtre[6]; // filtre pour scanf
                        sprintf(filtre, "%s%c%s","%d",SEPARATEUR,"%d"); //cree le filtre avec
le separateur


        sscanf(line.c_str()+i,filtre,(&players[player_cnt].level_IA),&(players[player_cnt].level_
multi)); //scan pour les deux valeurs dans la ligne

                        if(players[player_cnt].level_IA < 1 || players[player_cnt].level_IA > 7)
// si le niveau n'est pas correcte
                        {
                                cout << "JOUEUR COROMPU ! Mis au niveau IA 1 (" <<
players[player_cnt].level_IA << ")" << endl; // si le niveau joueur est faux
                                players[player_cnt].level_IA = 1; // on met le niveau a 1
                        }

                        if(players[player_cnt].level_multi < 1) // si le niveau n'est pas correcte
                        {
                                cout << "JOUEUR COROMPU ! Mis a 1 Nim point (" <<
players[player_cnt].level_multi << ")" << endl; // si le niveau joueur est faux
                                players[player_cnt].level_multi = 1; // on met le niveau a 1
                        }


                        player_cnt++; // on augmente le nombre de joueur

                        line = ""; //on vide la ligne
                }
        } while(!done);

        fclose(file); // on ferme le fichier

    }

    /* load_file
     *  But: ouvre et ecrit la liste des joueurs dans le fichier de sauvegarde
     */
    void write_file()
    {
        FILE * file; // le fichier qui represente le fichier
```

```
        file = fopen(NOM_FICHIER,"w"); // on ouvre le fichier pour ecrire dedans en effacant le
contenu
        assert(file != NULL); // si le fichier ne s'ouvre pas

        for(int i = 0 ; i < player_cnt ; i++) // sur tout les joueurs
        {
                char sep = SEPARATEUR; // le separateur
                fwrite(players[i].name.c_str(), 1, players[i].name.length(), file); // on ecrit le
nom dans le fichier
                fwrite(&sep, 1, 1, file); // on ecrit le separateur
                char buff[50];
                sprintf(buff, "%d%c%d",players[i].level_IA,SEPARATEUR,players[i].level_multi);
// on ecrit le nombre dans le tableau temporaraire
                fwrite(buff, 1, strlen(buff), file); // on ecrit le tableau temporaire
                fwrite(& "\n", 1, 1, file); // on ecrit une fin de ligne
        }

        fclose(file); // on ferme le fichier

    }

    /* return_base_info_for
     *  Entree: name= nom du joueur
     *  Sortie: player_info
     *   But: Retourne un player_info avec les valeurs de base
     */
    player_info return_base_info_for(string name)
    {
        player_info info; //nouvelles info
        info.level_IA = 1; //debut a 1
        info.level_multi = 1; //debut a 1
        info.name = name; // nouveau nom
        return info;
    }

    /* get_player_info
     *  Entree: name= nom du joueur
     *  Sortie: player_info
     *   But: Retourne les player_info d'un joueur
     */
    player_info get_player_info(string name)
    {
        player_info blank; // nom blank
        blank.name = "";

        for(int i = 0 ; i < player_cnt ; i++) // pour tout les joueurs
        {
                if(players[i].name == name) // si le nom correspond
                {
                        return players[i]; // on retourne le niveau du joueur
                }
        }
```

```
        return blank; //niveau par defaut
    }

    /* set_player_info
     *  Entree: name= nom du joueur, player_info= info
     *  Sortie: vrai = erreur
     *  But: definit les nouveaux player_info d'un joueur
     */
    bool set_player_info(string name, player_info info)
    {
        for(int i = 0 ; i < player_cnt ; i++) //pour tout les joueurs
        {
                if(players[i].name == name) // si le nom corresponds
                {
                        players[i] = info; // on change le niveau
                        return false;// on quitte la fonction
                }
        }

        return true;
    }

    /* add_player_info
     *  Entree: name= nom du joueur, player_info= info
     *  But: ajoute un nouveau joueur avec ses info dans la liste des joueurs
     */
    void add_player_info(string name, player_info info)
    {
        //si le joueur n'est pas trouve
        players = (player_info *) realloc(players, (player_cnt+1)*sizeof(player_info)); // on
rajoute un joueur
        new (players+player_cnt) player_info(); // on initialise le nouveau joueur
        players[player_cnt] = info; // on definit les info
        player_cnt++; // on ajoute un joueur
    }

    /* vider_ecran
     *  But: vider l'ecran
     */
    void vider_ecran()
    {
        for(int i = 0 ; i < 30 ; i++) // 30 fois
        {
                cout << endl;
        }
    }

    /* tracer_trait
     *  But: tracer une ligne de separation
     */
    void tracer_trait()
    {
```

```
    for(int i = 0 ; i < 20 ; i++) // 20 fois
    {
            cout << "-" << endl;
    }
}

/* texte_dans_boite
 *  Entree: texte= texte a ecrire
 *  But: tracer une boite et ecrire un texte dedans
 */
void texte_dans_boite(string texte)
{
    //ligne du haut
    cout << "          ";
    for(int i = 0 ; i < texte.length()+4 ; i++) // autant de fois que le texte est long + 4
    {
            cout << "#";
    }
    cout << endl;

    //ligne du milieu
    cout << "          # " << texte << " #" << endl;

    //ligne du bas
    cout << "          ";
    for(int i = 0 ; i < texte.length()+4 ; i++) // autant de fois que le texte est long + 4
    {
            cout << "#";
    }
    cout << endl;
}

/* texte_perdu
 *  Entree: nom_perdant= nom du joueur perdant
 *  But: afficher l'animation dans le cas ou un joueur perd
 */
void texte_perdu(string nom_perdant)
{
    vider_ecran();
    cout << "          =" << nom_perdant << " viens de piocher dans un tas vide !=" << endl;
    sleep(1); // on attends 1 seconde
}

/* texte_gagne
 *  Entree: nom_gagnant= nom du joueur gagnant
 *  But: afficher l'animation dans le cas ou un joueur gagne
 */
void texte_gagne(string nom_gagnant)
{
    vider_ecran();
    cout << "          =" << nom_gagnant << " viens de prendre le dernier baton !=" << endl;
    sleep(1); // on attends 1 seconde
```

```
    }

    /* set_player_info
     *  Entree: nom= nom du joueur, mode_IA= vrai->jeu contre l'IA
     *  Sortie: vrai = erreur
     *   But: definit les nouveaux player_info d'un joueur
     */
    void bienvenue_joueur(string nom, bool mode_IA)
    {
        player_info info = get_player_info(nom); // recupere les infos


        if(info.name != "") // si le nom n'est pas vide
        {
                cout << "> " << nom;
                if(mode_IA) // si en mode IA
                {
                        cout << ", Reprise au niveau " << info.level_IA << endl;
                }
                else
                {
                        cout << ", Vous avec " << info.level_multi << " Nim "<<
(info.level_multi>1?"points":"point") << endl;
                }
        }
        else // si le joueur n'as pas de niveau enregistre
        {
                info = return_base_info_for(nom); // on cree un nouveau joueur
                add_player_info(nom,info); // on ajoute le joueur

                cout << "> Bienvenue " << nom;
                if(mode_IA) // si l'IA est activee
                {
                        cout << ", Vous commencez au niveau " << info.level_IA << endl;
                }
                else
                {
                        cout << ", Vous commencez avec " << info.level_multi << " Nim " <<
(info.level_multi>1?"points":"point") << endl;
                }
        }
    }


    /*
     * main
     *
     *  But: fonction centrale
     */
    int main(int argc, char **argv)
    {
       //Initialisation du generateur aléatoire
```

```
	srand(time(NULL));

	 //demande les noms
	string nom[2];
	nom[0] = demander_texte("Quel sera le nom du joueur 1 ? (sans espaces)");
	nom[1] = demander_texte("Quel sera le nom du joueur 2 ? Ou \"IA\" pour activer le
robot (sans espaces)" );
	 bool activer_IA = (nom[1] == "IA" || nom[1] == "ia"); // on active si le joueur a repondu
IA


	 //stockage des scores
	 int score[2]={0,0};

	 //numero du joueur actuelle
	int num_joueur = 0;

	 //dernier tas dans lequel a pioché le joueur
	 int dernier_tas_joueur;

	 cout << endl;


	 load_file(); // on lit le fichier des joueurs
	 for(int i = 0 ; i < (activer_IA?1:2) ; i++)
	 {
		bienvenue_joueur(nom[i], activer_IA);
	 }

	 //niveau de l'IA
	 int niveau_IA = 0;
	 if(activer_IA)
	 {
		niveau_IA = get_player_info(nom[0]).level_IA;
	 }


	 cout << endl;
	 //maximum de batons qui peuvent être pris en un coup
	 int max_batons_par_coup = demander_nombre(0,-1,"[" + nom[0] + "] limiter a
combien de batons par coup? (0 pour illimite)");


	 cout << endl;
	 //maximum de batons par tas au debut du jeu
	int max_batons_par_tas = demander_nombre(1,-1,"[" + nom[0] + "] limiter a combien
de batons par tas ?");



	do //boucle des parties
	{
```

```
                //animation de debut de partie
                vider_ecran();
                texte_dans_boite(nom[num_joueur] + " commence la partie !");
                sleep(1); // on attends 2 secondes



        //Géneration de trois valeurs aléatoires dans un tableau
        int tas[3] = { rand()%max_batons_par_tas+1,rand()%max_batons_par_tas+1,
rand()%max_batons_par_tas+1};

                //permet de signaler la victoire
        bool gagne = false;
        while(!gagne) //boucle des tours de jeu
        {
                        vider_ecran(); // on vide l'ecran au debut du tour

                        // affiche la table de jeu
          for (int i = 0; i < 3 ; i++)
          {
            cout << "  -" << (char) ('A'+i)  << ": "; // affiche A B ou C grace à l'ascii
            for (int nba = 0; nba<tas[i] ; nba++) // boucle pour afficher tout les batons
            {
              cout << "|" ;
            }
            cout << endl;
          }

                        cout << endl << "Au tour de  -> " << nom[num_joueur] << "       -
        ";
                        if(max_batons_par_coup > 0) // si il y a une limite de batons par coup
                        {
                                cout << "maximum de batons par coup : " <<
max_batons_par_coup << endl;
                        }
                        else
                        {
                                cout << "pas de maximum de batons par coup" << endl;
                        }
                        cout << endl;

                        // => c'est à l'IA de jouer
                        if(activer_IA && num_joueur ==1) //si IA actif et tou du joueur 1 (aka le
second)
                        {
                                //animation pour l'IA
                                cout << "... L'IA joue ..." << endl;
                                sleep(1); // on attends 2 secondes

                                int numtas;       // numero du tas qui va être joué
                                int numenlev; // numero de batons qui vont être enlevés
```

```
                                bool ne_pas_finir_tas = false; // parametre qu'il ne faut pas
finir le tas


                                if(niveau_IA == 1) // niveau le plus simple
                                {
                                        //aléatoire pour le tas et le nombre de batons
                                        numtas = rand_juste(0,2);
                                        if(tas[numtas] != 0) // si il reste des batons
                                        {
                                                numenlev = rand_juste(1,tas[numtas]);
                                                if(max_batons_par_coup != 0 && numenlev >
max_batons_par_coup) // si il y a une limite de batons par coup et qu'on la depasse
                                                {
                                                        numenlev = max_batons_par_coup; //
enleve le max autorise par la limite
                                                }
                                        }
                                }
                                else if(niveau_IA >= 2) // niveau très facile
                                {
                                        // prend un tas au hasard qui N'EST PAS VIDE
                                        do
                                        {
                                                numtas = rand_juste(0,2);
                                        } while(tas[numtas] == 0);

                                        // definit le max de batons à 3 sauf si le tas n'a pas
assez de battons
                        int max_enlev = min(tas[numtas],3);

                                        // prend un nombre aléatoire entre 1 et le max ci-
dessus
                                        numenlev = rand_juste(1,max_enlev
);
                                        if(max_batons_par_coup != 0 && numenlev >
max_batons_par_coup) // si il y a une limite de batons par coup et qu'on depasse la limite
                                        {
                                                numenlev = max_batons_par_coup; // on
prend le max autorise
                                        }

                                }
                                if(niveau_IA >= 3) //niveau stratégique
                                {
                                        //limite pour qu'un tas soit considéré comme très plein
                                        const int limite_beaucoup_restant =
max(max_batons_par_tas/4,4);

                                        //nombre de batons max qui peuvent être retirés si
très plein
```

```
                                        const int retrait_beaucoup_restant =
max(max_batons_par_tas/4,3);

                                        //nombre de batons max qui peuvent être retirés
normalement
                                        const int retrait_normal =
max(max_batons_par_tas/4,10);

                                        //nombre de tas vide
                                        int nombre_tas_vide=0;
                                        for(int i = 0 ; i < 3 ; i++) //compte les tas vide
                                        {
                                                if(tas[i] == 0) // si le tas est vide
                                                {
                                                        nombre_tas_vide++;
                                                }
                                        }

                                        //nombre de tas qui n'a qu'un baton
                                        int nombre_tas_a_un=0;
                                        for(int i = 0 ; i < 3 ; i++) //compte les tas a un
                                        {
                                                if(tas[i] == 1) // si le tas n'a plus qu'un baton
                                                {
                                                        nombre_tas_a_un++;
                                                }
                                        }

                                        if(nombre_tas_vide == 1 && niveau_IA >= 5) // si 1 tas
vide, deux tas restants et que l'IA est niveau 5 minimum
                                        {
                                                numtas = tas_avec_mini_et_diff(tas,2,-1); //
cherche un tas avec plus de 1 batons
                                                if(numtas == -1) // pas de tas avec plus de 1
batons
                                                {
                                                        //ligne de debbogage, cette ligne est
cachee par le defilement.
                                                        cout << "Debug: forced !" << endl;
                                                   numtas = tas_avec_mini_et_diff(tas,1, -1);
//cherche un tas avec mini 1 baton
                                                }

                                                //ligne de debbogage, cette ligne est cachee
par le defilement.
                                                cout << "Debug: smallest not empty attack id "
<< numtas << endl;
                                                ne_pas_finir_tas = true; // essaye de ne pas
finir le tas
                                        }
                                        else if(nombre_tas_vide == 2 && niveau_IA >= 4) //si 2
tas vide, 1 tas restant et que l'IA est niveau 4 minimum
```

```
                                        {
                                                numtas = tas_avec_mini_et_diff(tas,2,-1); //
cherche un tas avec plus de 1 batons
                                                if(numtas == -1) // pas de tas avec plus de 1
batons
                                                {
                                                        //ligne de debbogage, cette ligne est
cachee par le defilement.
                                                        cout << "Debug: forced !" << endl;
                                                  numtas = tas_avec_mini_et_diff(tas,1,-1);
//cherche un tas avec mini 1 baton
                                                }
                                                //ligne de debbogage, cette ligne est cachee
par le defilement.
                                                cout << "Debug: last pile attack " << numtas <<
endl;
                                        }
                                        else if(nombre_tas_vide == 0 && nombre_tas_a_un ==
2 && niveau_IA >= 7) // si les trois tas restent, deux tas sont a un et que l'IA est niveau 7
                                        {
                                                numtas = tas_avec_mini_et_diff(tas,2,-1); // on
prend le tas qui n'est pas a 1
                                                //ligne de debbogage, cette ligne est cachee
par le defilement.
                                                cout << "Debug: leave two times one attack"
<< endl;
                                        }
                        else // si 0 tas vide, 3 tas restants
                                        {
                                                //ligne de debbogage, cette ligne est cachee
par le defilement.
                                                cout << "Debug: lowering" << endl;
                                                numtas = tas_le_plus_haut_bas(tas,false); //
cherche le tas avec le plus de batons
                                        }

                                        int numenlev_voulu; // nombre de batons à enlever
voulu

                                        if(nombre_tas_vide == 2 && niveau_IA >= 6) // 2 tas
vide, 1 tas restant et que l'IA est niveau 6 minimum
                                        {
                                                numenlev_voulu = 99999; // on veut tout
prendre
                                                //ligne de debbogage, cette ligne est cachee
par le defilement.
                                                cout << "Debug: cut short attack" << endl;
                                        }
                                        else if(nombre_tas_vide == 0 && nombre_tas_a_un ==
2 && niveau_IA >= 7) // si les trois tas restent, deux tas sont a un et que l'IA est niveau 7
                                        {
                                                numenlev_voulu = 99999;
```

```
                                        //ligne de debbogage, cette ligne est cachee
par le defilement.
                                        cout << "Debug: two times one attack PT2" <<
endl;
                                }
                                else // 1 ou 0 tas vide
                                {
                                        int min_enlev = 1; //valeur minimal qu'on peut
enlever
                                        int max_enlev = retrait_normal; //valeur
maximale qu'on peut enlever

                                        if(tas[numtas] > limite_beaucoup_restant &&
niveau_IA >= 4) // si il reste beaucoup de batons
                                        {
                                                //ligne de debbogage, cette ligne est
cachee par le defilement.
                                                cout << "Debug: fast drop down" <<
endl;
                                                max_enlev =
retrait_beaucoup_restant; // nouvelle valeur maximale qu'on peut enlever
                                        }
                                        //ligne de debbogage, cette ligne est cachee
par le defilement.
                                   cout << "Debug: rand " << min_enlev << " to " <<
max_enlev << endl;
                                   numenlev_voulu =
rand_juste(min_enlev,max_enlev); // on essaye de prendre un nombre de batons aléatoire
                                }


                                if(ne_pas_finir_tas && numenlev_voulu >= tas[numtas]
&& tas[numtas] > 1) // si il ne faut pas finir le tas, que l'on viderais le tas et qu'on en prend
plus d'un
                                {
                                        //ligne de debbogage, cette ligne est cachee
par le defilement.
                                        cout << "Debug: do not finish" << endl;


                                        numenlev_voulu = tas[numtas] - 1; // on prend
un baton de moins qu'il n'y en as, pour en laisser un
                                }

                                numenlev = min(numenlev_voulu,tas[numtas]); // on
prend le plus petit de ce qui est voulu et de ce qui est disponible
                                if(max_batons_par_coup > 0 && numenlev >
max_batons_par_coup) //si il y a un max de battons et qu'on en enlève plus que la limite
                                {
                                        numenlev = max_batons_par_coup; //on
enleve le max de batons.
                                }
```

```
                                        //ligne de debbogage, cette ligne est cachee par le
defilement.
                                        cout << "Debug: " << numenlev_voulu << " " <<
tas[numtas] << " " << max_batons_par_coup <<endl;
                                }


                                if(tas[numtas] == 0)// si le tas est vide
                                {
                                        texte_perdu(nom[num_joueur]);
                                        gagne=true; //un un joueur a gagné (le joueur actuel
viens de perdre, gardé pour correspondre au sujet)
                                        continue; // on saute la fin de la boucle car on a gagné
                                }


                                tas[numtas] -= numenlev; // on enlève les batons du tas

                                //ligne de debbogage, cette ligne est cachee par le defilement.
                                cout << endl << "Debug: L'IA \"" << nom[1]  << "\" a pris " <<
numenlev << (numenlev>1?" battons":" batton") << " sur le tas " << (char) ('A'+numtas) <<
endl;

                        }
                        else
                        {
                                int numtas; //numéro du tas choisis
                                char numtaschar = demander_lettre_sanscasse('A','C',"Saisie le
tas A,B ou C: "); // on demande la lettre du tas choisis
                                if(numtaschar < 'a') // la lettre est dans la zone ascii des
majuscules
                                {
                                        numtas = numtaschar - 'A'; // on transforme la valeur
ASCII du char en numéro
                                }
                                else // si la lettre est dans les minuscules
                                {
                                        numtas = numtaschar - 'a'; // on transforme la valeur
ASCII du char en numéro
                                }

                                if(tas[numtas] == 0) // si le tas est vide (dupliqué, pourait etre
remplacée par une fonction, mais garder pour correspondre à l'énoncé)
                                {
                                        texte_perdu(nom[num_joueur]);
                                        gagne=true; // un joueur a gagné (le joueur actuel
viens de perdre, gardé pour correspondre au sujet)
                                        continue; // on saute la fin de la boucle car on a gagné
                                }

                                int max_enlev = tas[numtas]; // nombre max de batons que le
joueur peut enlever
```

```
                                if(max_batons_par_coup > 0 && max_enlev >
max_batons_par_coup) // si il y a une limite de battons par coup et qu'elle est depasee
                                {
                                        max_enlev = max_batons_par_coup; // on limite le
max de batons que le joueur peut prendre
                                }
                                int numenlev = demander_nombre(1,max_enlev, "Combien
souhaitez vous en enlever ?"); // on demande le nombre de batons à enlever

                                dernier_tas_joueur = numtas; // on sauvegarde le dernier tas
joué pour l'IA
                                tas[numtas]-=numenlev; // on retire les batons
                        }


        if(tas[0]==0 && tas[1]==0 && tas[2]==0) // si les trois tas sont vides
        {
           gagne = true; // un joueur a gagné
                                texte_gagne(nom[num_joueur]);
        }

        num_joueur = !num_joueur; // on passe d'un joueur à l'autre (0->1->0->1)


            }
            /* En sortant de cette boucle, num_joueur est TOUJOURS LE PERDANT*/



            //animation de victoire
            sleep(1); // on attends 2 secondes
            vider_ecran();
            if(activer_IA) // si l'IA est active
            {
                    if(num_joueur == 1) // et que le joueur qui viens de perdre est le
numero 1 (aka le second)
                    {
                            texte_dans_boite("Victoire !!");
                    }
                    else
                    {
                            texte_dans_boite("Defaite.");
                    }
            }
            else // si il s'agit d'un joueur contre joueur
            {
                    texte_dans_boite(nom[!num_joueur] + " a vaincu " +
nom[num_joueur] + " !");
            }

            cout << endl << endl << endl;
```

```
		score[!num_joueur]++; // on augmente le score du vainqueur

			if(activer_IA)
			{
				if(num_joueur == 1) // si l'IA est active et que le joueur a gagne
				{
					if(niveau_IA < 7) // tant que le joueur n'a pas atteint le niveau
max
					{
						niveau_IA++; // on augmente le niveau de l'IA
						cout << "Vous etes monte d'un niveau ! (vous etes
desormais niveau " << niveau_IA << ")" << endl;
					}
					else // si le joueur est deja au max
					{
						cout << "Vous avez deja atteins le niveau maximum !"
<< endl;
					}
				}
				else
				{
					if(niveau_IA > 1) // tant que le joeur n'a pas atteint le niveau
min
					{
						niveau_IA--; // on augmente le niveau de l'IA
						cout << "Vous etes descendu d'un niveau ! (vous etes
desormais niveau " << niveau_IA << ")" << endl;
					}
					else
					{
						cout << "Vous ... restez niveau 1 !" << endl;
					}
				}
				player_info info = get_player_info(nom[0]); // on recupere les infos
joueur
				assert(info.name != ""); // si le joueur n'existe pas

				info.level_IA = niveau_IA; // on change le niveau de l'IA
				set_player_info(nom[0], info); // on met a jour les info utilisateurs
			}
			else
			{
				for(int i = 0 ; i < 2 ; i++) // pour les deux joueurs
				{
					player_info info = get_player_info(nom[i]); // on recupere les
info
					assert(info.name != ""); // si le joueur n'existe pas

					if(num_joueur == i && info.level_multi > 1) // si le joueur viens
de perdre et qu'il n'est pas au niveau minimal
					{
```

```
                                        info.level_multi--;
                                }
                                if(num_joueur != i) // si le joueur viens de gagner
                                {
                                        info.level_multi++;
                                }
                                set_player_info(nom[i], info); // on met a jour les infos des
joueurs
                        }
                }
                write_file(); // on enregistre les joueurs dans la base de donnees

                cout << endl;
        }
        while(demander_bool(nom[0] + (activer_IA?"":" et "  + nom[1]) + ", souhaitez vous
rejouer ?")); // tant que le joueur répond oui à la question ci-dessus

         cout << endl << endl << "Resume des victoires: " << nom[0] << " a remporte " <<
score[0] << (score[0]>1?" victoires et ":" victoire et ") << nom[1] << " a remporte " << score[1]
<< (score[1]>1?" victoires.":" victoire.") << endl;
        if(!activer_IA) // si l'IA n'est pas activer
        {
                int score0 = get_player_info(nom[0]).level_multi; // score du joueur 0
                int score1 = get_player_info(nom[1]).level_multi; // score du joueur 1
                cout << endl << endl << nom[0] << " a donc " << score0 << (score0>1?" Nim
points et ":" Nim point et ") << nom[1] << " a " << score1 << (score1>1?" Nim points.":" Nim
point.") << endl;
        }
    }
```