```cpp
#include <iostream>
#include <string.h>
#include <string>
#include <ios>
#include <iomanip>
#include <random>

using namespace std;


//グローバル変数
int CNT;

//プレイヤー(キャラ)の情報格納用構造体
struct MEMBER{
  char Name[16];  //プレイヤー名
  int  Type;      //0なら人,1ならCOM
  int  Strength;  //COMの強さ(=dep)
};

//プロトタイプ宣言
bool GetPlayer(int *pos, int END, int array[][32], int times, MEMBER member[3], int Pnum);
bool GetAI(int *pos, int END, int array[][32], int times, MEMBER member[3], int Pnum);
void GetOperate(int pos, int END, int dep);
void GetMemory(int array[][32], int n, MEMBER member[3]);
int random(int s, int e);

int main(void){

  int pos = 0;               //現状態
  int endPoint;              //ゲームの終点
  int style;                 //プレイ人数選択用
  int level;                 //レベル選択用
  int turn = 0;              //ターン数
  int memory[3][32] = {};    //ゲームログ
  int order;                 //プレイ順序

  MEMBER member[3] = {};     //MEMBER構造体の宣言
  MEMBER temporary[1] = {}; //入れ替えようの一時保管変数
  MEMBER *p;    //操作用ポインタ
  MEMBER *tmp;  //同上
  p = member;       //ポインタ割り当て
  tmp = temporary;  //同上

  std::cout << '\n' << "##GameSetting##" << '\n';

  //プレイ人数選択
  std::cout << "Please select the number of players" << '\n';
  std::cout << "1) Human : 0 | COM : 3 " << '\n';
  std::cout << "2) Human : 1 | COM : 2 " << '\n';
  std::cout << "3) Human : 2 | COM : 1 " << '\n';
  std::cout << "4) Human : 3 | COM : 0 " << '\n';
  while (1) {
    std::cout << ">> ";
    std::cin >> style;
    if (style > 0 && style < 5) {
      std::cout << '\n';
      break;
    }
    std::cout << "!! Please choose a number from 1 to 4 !!" << '\n';
  }

  //プレーヤー名入力
  for (size_t i = 0; i < style-1; i++) {
    std::cout << "Please enter the name of Player_" << i+1 << '\n' << ">> ";
    std::cin >> (p+i)->Name;
  (p+i)->Type = 0;
}
//COM名入力
for (size_t i = style; i < 4; i++) {
  std::cout << "Please enter the name of COM_" << i-(style-1) << '\n' << ">> ";
  std::cin >> (p+i-1)->Name;
  (p+i-1)->Type = 1;
}

//ゲームの終了点の選択
std::cout << '\n' << "Please select the end point of the game" << '\n' << ">> ";
std::cin >> endPoint;

//COMのレベル選択
std::cout << '\n';
for (size_t i = style; i < 4; i++) {
  while (1) {
    std::cout << "Please choose the strength of the " << (p+i-1)->Name << '\n';
    std::cout << "1:weak | 2:middle | 3:strong " << '\n' << ">> ";
    std::cin >> level;
    if (level > 0 && level <4) {
      break;
    }
    std::cout << "!! Please choose a number from 1 to 3 !!" << '\n';
  }
  if (level == 1) {
    //1手先読み
    (p+i-1)->Strength = 2;
  }else if (level == 2) {
    //4手先読み
    (p+i-1)->Strength = 8;
    //depが偶数になるよう調整
    if ((p+i-1)->Strength % 2 != 0) {
      (p+i-1)->Strength += 1;
    }
  }else if (level == 3) {
    //全手先読み
    (p+i-1)->Strength = endPoint/2;
    //depが偶数になるよう調整
    if ((p+i-1)->Strength % 2 != 0) {
      (p+i-1)->Strength += 1;
    }
  }
}

//プレイ順の選択
std::cout << '\n' << "Decide the order of play at random" << '\n';
order = random(0, 5);

//プレイ順に構造体を入れ替え
switch (order) {
  case 0:
  //0->1->2
  break;

  case 1:
  //1->2->0
  *tmp = *p;
  *p = *(p+1);
  *(p+1) = *tmp;
  *tmp = *(p+1);
  *(p+1) = *(p+2);
  *(p+2) = *tmp;
  break;

  case 2:
  //2->0->1
  *tmp = *p;
  *p = *(p+1);
  *(p+1) = *tmp;
  *tmp = *p;
  *p = *(p+2);
  *(p+2) = *tmp;
  break;

  case 3:
  //1->0->2
  *tmp = *p;
  *p = *(p+1);
  *(p+1) = *tmp;
  break;

  case 4:
  //0->2->1
  *tmp = *(p+1);
  *(p+1) = *(p+2);
  *(p+2) = *tmp;
  break;

  case 5:
  //2->1->0
  *tmp = *p;
  *p = *(p+2);
  *(p+2) = *tmp;
  break;
}

//プレイ順の表示
std::cout << "1st : " << p->Name << '\n';
std::cout << "2nd : " << (p+1)->Name << '\n';
std::cout << "3rd : " << (p+2)->Name << '\n';

//メインループ開始
std::cout << '\n' << "##GameStart##" << '\n';
while (1){
  //1番手
  if (p->Type == 0) {  //構造体のType部で人かCOMか判断
    if ( GetPlayer(&pos, endPoint, memory, turn, member, 0) ){
      GetMemory(memory, turn, member);
      break;
    }
  }else{
    if ( GetAI(&pos, endPoint, memory, turn, member, 0) ){
      GetMemory(memory, turn, member);
      break;
    }
  }

  //2番手
  if ((p+1)->Type == 0) {
    if ( GetPlayer(&pos, endPoint, memory, turn, member, 1) ){
      GetMemory(memory, turn, member);
      break;
    }
  }else{
    if ( GetAI(&pos, endPoint, memory, turn, member, 1) ){
      GetMemory(memory, turn, member);
      break;
    }
  }

  //3番手
  if ((p+2)->Type == 0) {
    if ( GetPlayer(&pos, endPoint, memory, turn, member, 2) ){
      GetMemory(memory, turn, member);
      break;
    }
  }else{
    if ( GetAI(&pos, endPoint, memory, turn, member, 2) ){
      GetMemory(memory, turn, member);
      break;
    }
  }

```

```
    //現在状態の表示
    GetMemory(memory, turn, member);
    turn++;
  }
  return 0;
}

//プレイヤーの入力関数
bool GetPlayer(int *pos, int END, int array[][32], int times, MEMBER member[3], int Pnum){

  int choice;

  std::cout << '\n' << "Now number is " << *pos << '\n';

  while (1){
    if (*pos < END-1) {
      std::cout << "Please push number (1 or 2) " << member[Pnum].Name << '\n';
      std::cout << "-> ";
      std::cin >> choice;
      if (choice == 1 || choice == 2) {
        break;
      }
      std::cout << "!! You can only enter 1 or 2 !!" << '\n';
    }else if(*pos == END-1){
      std::cout << "Please push number 1 " << member[Pnum].Name << '\n';
      std::cout << "-> ";
      std::cin >> choice;
      if (choice == 1) {
        break;
      }
      std::cout << "!! You can only enter 1 !!" << '\n';
    }
  }

  //現状態を更新しログを保存
  *pos += choice;
  array[Pnum][times] = *pos;

  if (*pos >= END){
    std::cout << member[Pnum].Name << " Lose..." << '\n';
    return true;
  }

  return false;
}

//AIの選択関数
bool GetAI(int *pos, int END, int array[][32], int times, MEMBER member[3], int Pnum){

  int one, two;  //1,2それぞれの勝率を格納
  int operate;   //最終的な決定
  int dep = member[Pnum].Strength;  //COMの強さごとに決められた探索深度

  std::cout << '\n' << member[Pnum].Name << " thiking now..." << '\n';
  //先読み
  for (size_t i = 1; i < 3; i++) {
    //AIが 1or2 を選んだ場合の勝ち数を計測
    CNT = 0;
    GetOperate(*pos+i, END, dep);
    if (i == 1){
      one = CNT;
    }else if(i == 2){
      two = CNT;
    }
  }

  if (*pos > END-3) {
    operate = 1;
  }else if (*pos < END/2) {
    operate = random(1, 2);
  }else{
    //勝率が高い方を選択
```

```cpp
    if (one > two) {
      operate = 1;
    }else if (one < two){
      operate = 2;
    }else{
      //どちらでも同じならランダムで決定
      operate = random(1,2);
    }
  }

  //現状態を更新しログを保存
  *pos += operate;
  array[Pnum][times] = *pos;

  std::cout << member[Pnum].Name << " selected " << operate << '\n';

  if (*pos >= END){
    std::cout << member[Pnum].Name << " Lose..." << '\n';
    return true;
  }
  return false;
}

//AIのオペレーター関数
void GetOperate(int pos, int END, int dep){
  //再帰を繰り返し指定の深さまできたら終了
  if (dep == 0) {
    return;
  }

  if (dep % 2 == 0) {
    //depが偶数=人間のターン
    for (size_t i = 2; i < 5; i++) {
      //posが終了点を超えていたら終了
      if (pos + i >= END-2) {
        return;
      }
      GetOperate(pos+i, END, dep-1);
    }
  }else{
    //depが奇数=AIのターン
    for (size_t i = 1; i < 3; i++) {
      if (pos + i >= END-2) {
        //終了点の数値があればグローバル変数CNTをインクリメント
        if (pos + i == END-2) {
          CNT++;
        }
        return;
      }
      GetOperate(pos+i, END, dep-1);
    }
  }
}

//ゲームログの表示保存関数
void GetMemory(int array[][32], int n, MEMBER member[3]){

  std::cout << '\n' << "##################";
  for (size_t i = 0; i < n; i++) {
    std::cout << "####";
  }
  std::cout << '\n';

  for (size_t i = 0; i < 3; i++) {

    if (i == 0) {
      std::cout << std::left << std::setw(10) << member[0].Name << "...";
    }else if (i == 1){
      std::cout << std::left << std::setw(10) << member[1].Name << "...";
    }else if (i == 2){
      std::cout << std::left << std::setw(10) << member[2].Name << "...";
    }

    for (size_t j = 0; j < n+1; j++) {
      if (array[i][j] != 0){
        std::cout << std::right << std::setw(3) << array[i][j];
      }else{
        std::cout << std::right << std::setw(3) << "--";
      }
      if(j == n){
        std::cout << " ";
      }else{
        std::cout << ",";
      }
    }
    std::cout << '\n';
  }

  std::cout << "##################";
  for (size_t i = 0; i < n; i++) {
    std::cout << "####";
  }
  std::cout << '\n';
}

//乱数生成
int random(int s, int e){
  std::random_device rnd;     // 非決定的な乱数生成器を生成
  std::mt19937 mt(rnd());     //  メルセンヌ・ツイスタの32ビット版、引数は初期シード値
  std::uniform_int_distribution<> r(s, e);        // 指定範囲の一様乱数
  return r(mt);
}

```