Université de Nice-Sophia Antipolis — POLYTECH
ELSE4-FISE — 2024–2025

Examen de C++

Durée : 1h00
Aucun document autorisé
Mobiles interdits

▶ 1. Citez et décrivez 4 qualités fondamentales recherchées dans la conception et le développement d'un logiciel.

---

voir cours

.....

▶ 2. Dans la déclaration d'une méthode d'une classe C++, à quoi correspond l'utilisation du mot-clé **const** à la fin de son prototype ?

---

voir cours

.....

On souhaite obtenir la représentation binaire en complément à 2 de nombres entiers décimaux de type **long long int** sur un nombre variable de bits. Par exemple, sur 8 bits, la représentation binaire de l'entier décimal 20 est 00010100, et celle de -114 est 10001110. De même, sur 16 bits, on a $1024 = 0000010000000000$ et $-1 = 1111111111111111$. Le nombre de bits pour la représentation d'un nombre décimal pourra donc varier de 1 à `sizeof(long long int)*8`. D'autre part, l'entier à convertir sur b bits doit être compris entre -2^{b-1} et $2^{b-1}-1$. Pour les vérifications de validité, vous pourrez utiliser `assert`.

▶ 3. Dans le fichier `décimal.hpp`, écrivez la classe `décimal` avec
- la constante membre *privée* `maxBits` avec sa valeur ;
- la variable membre *privée* `n` de type **long long int** ;
- un constructeur *public* qui initialise la variable membre `n` ;
- l'en-tête de la méthode *publique* `getN` qui renvoie la valeur décimale de `n` ;
- l'en-tête de la méthode *publique* `versBinaire` qui renvoie sous forme d'une chaîne de caractères (`std::string`) la représentation binaire de la variable membre `n`. Le nombre de bits est passé en paramètre.

```
#pragma once
#include <string>

class décimal {
private:
  static const int maxBits = sizeof(long long int)*8;
  long long int n;
public:
  // le constructeur
  nombre(const long long int x) : n(x) {}
  // accesseur
```



```
  long long int getN() const;
  // conversion
  std::string versBinaire(const int nbBits) const;
};
```

.....

▶ 4. Dans le fichier `décimal.cpp`, programmez la méthode `versBinaire`. *Indication* : il s'agit de faire la décomposition de `this->n` en base 2. Notez que si `this->n` est négatif, il sera judicieux de décomposer son complément sur la plus grande valeur décimale sur le nombre de bits demandés. Par exemple, le complément de -114 sur 8 bits est $-114 + 256 = 142$. La décomposition binaire de 142 est 10001110, qui est donc la représentation binaire de -114 sur 8 bits. Pour créer la chaîne de caractères résultat renvoyée par la méthode `versBinaire`, utilisez l'opérateur `+` et la méthode `insert` de la classe `std::string` (voir annexe). Enfin, vous ne devez pas utiliser la fonction `pow` (ou toute autre fonction) pour calculer des puissances de 2.

```
/**
 * Antécédent : 0 < nbBits ⩽ this->maxBits
 * Rôle : cette méthode renvoie la représentation binaire sur nbBits
 *        sous forme d'une std::string du nombre décimal courant
 */
std::string nombre::versBinaire(const int nbBits) const {
  assert(nbBits>0 and nbBits<=this->maxBits);
  // le nombre de bits est valide
  std::string s="";
  if (this->n==0)
    return s.insert(0, nbBits, '0');
  // calculer le plus gd entier sur nbBits
  const long long int nbBitsP2 = 1LL<<nbBits;
  long long int _n = this->n;
  // vérifier que _n est dans les bornes
  // et prendre son complément à 2 s'il est négatif
  if (_n<0) {
      assert(_n>=-nbBitsP2/2);
      // prendre le complément de n à
      _n+=nbBitsP2;
  }
  else+
      assert(_n<nbBitsP2/2);
  // ok : n est dans les bornes ⇒ le décomposer en base 2
  // dans la chaîne de caractères s
  do
     s = std::to_string(_n%2) + s;
  while ((_n/=2)!=0);
  // compléter avec les bits de poids fort à 0
  return s.insert(0, nbBits-s.size(), '0');
}
```

.....

▶ 5. Écrivez la fonction `main` qui :
- lit sur l'E/S un nombre de bits ∈ [2, `sizeof(long long int)*8`].
- déclare la variable `table`, un vecteur de pointeurs sur `décimal`. La taille du vecteur est égale au nombre maximum de décimaux sur le nombre de bits lu ;
- initialise le vecteur avec les décimaux compris entre -2^{b-1} et $2^{b-1}-1$, où b est le nombre

de bits lu sur l'E/S

— parcourt toute la table pour écrire sur la S/S tous les décimaux avec, pour chacun, sa représentation binaire.

Par exemple, si on lit sur l'E/S l'entier 8, l'exécution de la fonction `main` écrira 256 nombres surt la S/S :

```
nb bits = 8
-128 = 10000000
-127 = 10000001
-126 = 10000010
....
  -2 = 11111110
  -1 = 11111111
   0 = 00000000
   1 = 00000001
....
 125 = 01111101
 126 = 01111110
 127 = 01111111
```

```
int main() {
  int nbBits;
  // lire le nb de bits sur l'E/S
  std::cout << "nb bits = ";
  std::cin >> nbBits;
  //
  assert(nbBits>0 and nbBits<=sizeof(long long int)*8);
  // déclarer la table
  long long int maxDécimaux = 1<<nbBits; // le nombre max de décimaux sur nbBits
  std::vector<décimal*> table(maxDécimaux);
  // créer la table des décimaux
  long long int n=-maxDécimaux/2;
  for (int i=0; i<maxDécimaux; i++, n++)
     table[i] = new décimal(n);
  // écrire la représentation binaire sur nbBits
  // de tous les décimaux contenus dans la table
  for (const décimal *n : table)
     std::cout << std::setw(nbBits/2) << n->getN() << " = "
                                      << n->versBinaire(nbBits) << std::endl;
  //
  return EXIT_SUCCESS;
}
```

.....