Примеры задач экзамена по ЭВМ

М. А. Ложников

1 Примеры задач на C++

Внимание. Этот список примерный, служит для иллюстрации уровня сложности задач. Сами задачи в билете могут отличаться.

Задача 1. *Дан класс комплексного числа.*

```
class Complex {
 public:
  Complex(double re, double im = 0) :
    re(re),
    im(im)
  { }

 private:
  double re;
  double im;
};

std::ostream& operator<<(std::ostream& out, const Complex& number);
```

Требуется перегрузить операцию сложения `+` *для двух классов* `Complex`*, перегрузить операцию вывода в поток и реализовать программу, иллюстрирующую пример использования операций.*

Задача 2. *Дан класс комплексного числа.*

```
class Complex {
 public:
  Complex(double re, double im = 0) :
    re(re),
    im(im)
  { }

 private:
  double re;
  double im;
};

std::ostream& operator<<(std::ostream& out, const Complex& number);
```

Требуется перегрузить операцию `+=` *для двух классов* `Complex`*, перегрузить операцию вывода в поток и реализовать программу, иллюстрирующую пример использования реализованных функций и операций.*

Задача 3. *Дан класс строки.*

```
class String {
 public:
  String(const char* str);

  ~String();

  char* data;
};

std::ostream& operator<<(std::ostream& out, const String& str);
```

Требуется реализовать указанные функции, а также перегрузить операцию `+` *для двух классов* `String` *и реализовать программу, иллюстрирующую пример использования реализованных функций и операций.*

Задача 4. *Дан класс строки.*

```
class String {
 public:
  String(const char* str);

  ~String();

  char* data;
};

std::ostream& operator<<(std::ostream& out, const String& str);
```

Требуется реализовать указанные функции, а также перегрузить операцию `+=` *для двух классов* `String` *и реализовать программу, иллюстрирующую пример использования реализованных функций и операций.*

Задача 5. *Дан класс строки.*

```
class String {
 public:
  String(const char* str);

  ~String();

  char* data;
};

std::ostream& operator<<(std::ostream& out, const String& str);
```

Требуется реализовать указанные функции, а также перегрузить константную и неконстантную операции `[]` *для чтения и изменения символов строки соответственно и реализовать программу, иллюстрирующую пример использования реализованных функций и операций.*

Задача 6. *Дан класс матрицы.*

```
class Matrix {
 public:
  Matrix(const double* data, int numRows, int numCols);

  Matrix(const Matrix& other);
  Matrix& operator=(const Matrix& other);

  ~Matrix();

  void Print() const;

 private:
  double* data;
  int numRows;
  int numCols;
};
```

Требуется реализовать указанные функции и реализовать программу, иллюстрирующую пример их использования.

Задача 7. *Дан класс матрицы.*

```
class Matrix {
 public:
  Matrix(int numRows, int numCols);

  Matrix(const Matrix& other) = delete;
  Matrix& operator=(const Matrix& other) = delete;

  double operator()(int row, int col) const;
  double& operator()(int row, int col);

  ~Matrix();

  void Print() const;

 private:
  double* data;
  int numRows;
  int numCols;
};
```

Требуется реализовать указанные функции и реализовать программу, иллюстрирующую пример их использования.

Задача 8. *Дан класс бинарного счётчика, который служит для представления неотрицательного двоичного числа.*

```
class BinaryCounter {
 public:
```

```
  BinaryCounter();

  BinaryCounter(const BinaryCounter& other) = delete;
  BinaryCounter& operator=(const BinaryCounter& other) = delete;

  ~BinaryCounter();

  void Print() const;

 private:
  // Массив хранит нули и единицы --- цифры двоичного числа.
  int* data;
  // Количество элементов в массиве.
  int count;
};
```

Требуется реализовать указанные функции, а также префиксную операцию инкремента для бинарного счётчика и реализовать программу, иллюстрирующую пример использования указанных функций и операций.

Задача 9. *Дан класс целых чисел по произвольному модулю.*

```
class Zk {
 public:
  Zk(int value, int modulus);

  int value;
  int modulus;
};
```

Требуется реализовать указанные функции, а также перегрузить операции сложения и умножения для класса `Zk` *и целого числа (операции возвращают класс* `Zk` *с тем же модулем) и реализовать программу, иллюстрирующую пример их использования.*

Задача 10. *Дан класс десятичных неотрицательных чисел.*

```
class Decimal {
 public:
  Decimal(int* digits, int count);

  Decimal(const Decimal& other);
  Decimal& operator=(const Decimal& other);

  ~Decimal();
  // Массив цифр от 0 до 9, задающих число.
  int* digits;
  // Их количество.
  int count;
};
```

Требуется реализовать указанные функции, а также перегрузить операцию сложения двух чисел типа `Decimal` *и реализовать программу, иллюстрирующую пример использования указанных функций и операций.*

2 Примеры задач на структуры данных

Внимание. Этот список примерный, служит для иллюстрации уровня сложности задач. Сами задачи в билете могут отличаться.

Задача 11. *В элементах односвязного списка типа* `ListNode` *записаны цифры числа в порядке от младшего разряда к старшему.*

```
struct ListNode {
  ListNode* next;
  int digit;
};

ListNode* Process(ListNode* headA, ListNode* headB);
```

Требуется написать процедуру сложения двух таких чисел.

Задача 12. *В элементах односвязного списка типа* `ListNode` *записаны некоторые числа.*

```
struct ListNode {
  ListNode* next;
  int value;
};

ListNode* Process(ListNode* head);
```

Требуется написать процедуру для циклического сдвига элементов списка. Процедура не должна менять значения в ячейках списка и создавать новые ячейки. Разрешается только переставлять ячейки местами.

Задача 13. *В элементах односвязного списка типа* `ListNode` *записаны некоторые числа.*

```
struct ListNode {
  ListNode* next;
  int value;
};

ListNode* Process(ListNode* head, int k);
```

Требуется написать процедуру для удаления k-ого элемента с конца списка. Процедура не должна менять значения в ячейках списка и создавать новые ячейки. Разрешается только переставлять ячейки местами.

Задача 14. *В элементах односвязного списка типа* `ListNode` *записаны некоторые числа, отсортированные в порядке возрастания.*

```
struct ListNode {
  ListNode* next;
  int value;
};

ListNode* Process(ListNode* head);
```

Требуется написать функцию, которая удаляет из отсортированного списка повторяющиеся элементы. Процедура не должна менять значения в ячейках списка и создавать новые ячейки. Разрешается только переставлять ячейки местами.

Задача 15. *В элементах односвязного списка типа* `ListNode` *записаны некоторые числа.*

```
struct ListNode {
  ListNode* next;
  int value;
};

ListNode* Process(ListNode* head);
```

Требуется написать функцию, которая переставляет элементы списка в обратном порядке. Процедура не должна менять значения в ячейках списка и создавать новые ячейки. Разрешается только переставлять ячейки местами.

Задача 16. *В элементах односвязного списка типа* `ListNode` *записаны некоторые числа.*

```
struct ListNode {
  ListNode* next;
  int value;
};

ListNode* Process(ListNode* head);
```

Требуется написать функцию, которая меняет местами соседние пары элементов списка, то есть для всех k меняет местами элементы с номерами $2k$ и $2k + 1$. Процедура не должна менять значения в ячейках списка и создавать новые ячейки. Разрешается только переставлять ячейки местами.

Задача 17. *В элементах односвязного списка типа* `ListNode` *записаны некоторые числа.*

```
struct ListNode {
  ListNode* next;
  int value;
};

bool Process(ListNode* head);
```

Требуется написать функцию, которая определяет, есть ли в списке цикл. Считается, что в списке есть цикл, если в одной из его ячеек указатель `next` *указывает на другую ячейку, расположенную ранее.*

Задача 18. *В элементах односвязного списка типа* `ListNode` *записаны некоторые числа.*

```
struct ListNode {
  ListNode* next;
  int value;
};

bool Process(ListNode* head);
```

Требуется написать функцию, которая определяет, является ли список палиндромом.

Задача 19. *В элементах односвязного списка типа* `List` *записаны некоторые числа.*

```
struct List {
  List* next;
  int value;
};

ListNode* Process(ListNode* head, int value);
```

Требуется написать функцию, которая удаляет из списка все элементы со значением `value`.

Задача 20. *В элементах односвязного списка типа* `List` *записаны некоторые числа.*

```
struct List {
  List* next;
  int value;
};

ListNode* Process(ListNode* head);
```

Требуется написать функцию, которая сортирует список при помощи сортировки пузырьком. Запрещается изменять значения в ячейках и создавать новые ячейки. Можно только переставлять ячейки местами.

Задача 21. *Для представления бинарного дерева используется следующая структура:*

```
struct TreeNode {
  int value;
  TreeNode* left;
  TreeNode* right;
};
```

Написать функцию, которая проверяет, является ли бинарное дерево деревом поиска.

```
bool Process(TreeNode* root);
```

Внимание. *Для функции* `Process()` *действуют следующие ограничения. В функции не должно быть операций ввода/вывода. Обычные массивы использовать нельзя. Контейнерами STL пользоваться нельзя. Нельзя менять значения в узлах дерева, можно только переставлять узлы местами. Задачу нужно решить наиболее оптимальным образом. В структуру* `TreeNode` *можно добавлять элементы при необходимости.*

Задача 22. *Для представления бинарного дерева используется следующая структура:*

```
struct TreeNode {
  int value;
  TreeNode* left;
  TreeNode* right;
};
```

Написать функцию, которая считает ширину дерева. Под шириной дерева подразумевается максимальное расстояние между узлами, не равными `NULL` *и расположенными на одном уровне. Если между двумя крайними узлами, расположенными на одном уровне, есть узлы, равные* `NULL`*, то они тоже учитываются при подсчёте расстояния.*

```
int Process(TreeNode* root);
```

Внимание. *Для функции* `Process()` *действуют следующие ограничения. В функции не должно быть операций ввода/вывода. Обычные массивы использовать нельзя. Контейнерами STL пользоваться нельзя. Нельзя менять значения в узлах дерева, можно только переставлять узлы местами. Задачу нужно решить наиболее оптимальным образом. В структуру* `TreeNode` *можно добавлять элементы при необходимости.*

Задача 23. *Для представления бинарного дерева используется следующая структура:*

```
struct TreeNode {
  int value;
  TreeNode* left;
  TreeNode* right;
};
```

Написать функцию, которая возвращает число узлов в бинарном дереве.

```
int Process(TreeNode* root);
```

Внимание. *Для функции* `Process()` *действуют следующие ограничения. В функции не должно быть операций ввода/вывода. Обычные массивы использовать нельзя. Контейнерами STL пользоваться нельзя. Нельзя менять значения в узлах дерева, можно только переставлять узлы местами. Задачу нужно решить наиболее оптимальным образом. В структуру* `TreeNode` *можно добавлять элементы при необходимости.*

Задача 24. *Для представления бинарного дерева используется следующая структура:*

```
struct TreeNode {
  int value;
  TreeNode* left;
  TreeNode* right;
};
```

В узлах дерева записаны цифры от 0 до 9. Таким образом, каждый путь от корня к листу задаёт число. Требуется написать функцию, которая считает сумму всех таких чисел.

```
int Process(TreeNode* root);
```

Внимание. *Для функции* `Process()` *действуют следующие ограничения. В функции не должно быть операций ввода/вывода. Обычные массивы использовать нельзя. Контейнерами STL пользоваться нельзя. Нельзя менять значения в узлах дерева, можно только переставлять узлы местами. Задачу нужно решить наиболее оптимальным образом. В структуру* `TreeNode` *можно добавлять элементы при необходимости.*

Задача 25. *Для представления бинарного дерева используется следующая структура:*

```
struct TreeNode {
  int value;
  TreeNode* left;
  TreeNode* right;
};
```

Написать функцию, которая проверяет, является ли дерево полным. Дерево называется полным, если все его слои, возможно, кроме последнего не имеют узлов, равных `NULL`*, а в последнем слое все узлы занимают наиболее левые позиции.*

```
bool Process(TreeNode* root);
```

Внимание. *Для функции* `Process()` *действуют следующие ограничения. В функции не должно быть операций ввода/вывода. Обычные массивы использовать нельзя. Контейнерами STL пользоваться нельзя. Нельзя менять значения в узлах дерева, можно только переставлять узлы местами. Задачу нужно решить наиболее оптимальным образом. В структуру* `TreeNode` *можно добавлять элементы при необходимости.*

Задача 26. *Для представления бинарного дерева используется следующая структура:*

```
struct TreeNode {
  int value;
  TreeNode* left;
  TreeNode* right;
};
```

Написать функцию, которая проверяет, является ли дерево чётно-нечётным. Дерево называется чётно-нечётным, если на чётных слоях расположены узлы с чётными значениями в порядке возрастания слева направо, а на нечётных слоях узлы с нечётными значениями в порядке убывания слева направо.

```
bool Process(TreeNode* root);
```

Внимание. *Для функции* `Process()` *действуют следующие ограничения. В функции не должно быть операций ввода/вывода. Обычные массивы использовать нельзя. Контейнерами STL пользоваться нельзя. Нельзя менять значения в узлах дерева, можно только переставлять узлы местами. Задачу нужно решить наиболее оптимальным образом. В структуру* `TreeNode` *можно добавлять элементы при необходимости.*