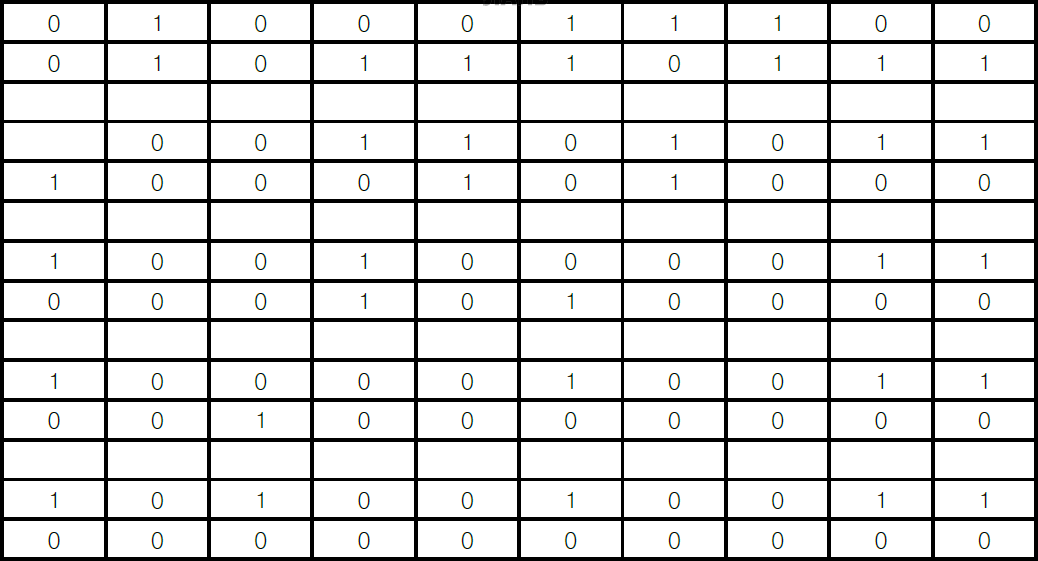
# Дисциплина «Алгоритмы решения прикладных задач» Рабочая тетрадь 3.

**Алгоритмы длинной арифметики: сложение, вычитание, умножение, деление**

|  |
| --- |
| **Теоретический материал** |
| Рассматриваемые алгоритмы применимы, прежде всего, к длинным и очень длинным целым числам (состоящими, например из 100 или 1000 цифр). Такие числа не могут быть записаны в переменную стандартных типов (int, long и т.п.). Поэтому программная реализация арифметических алгоритмов будет предполагать работу с массивом, каждый элемент которого будет представлять собой цифру в соответствующем разряде длинного числа. Массив можно создавать статически или динамически. Предпочтительный вариант – динамическое создание массивов (на языке C++ с помощью указателей). При этом в отдельной переменной необходимо хранить размер массива (количество цифр в длинном числе).  *Пример 1 демонстрирует создание динамического массива для числа, содержащего задаваемое с клавиатуры количество цифр. Массив заполняется* ***случайными*** *цифрами от 0 до 9 (старший разряд не может быть нулевым в этом случае). При этом в нулевом элементе массива содержится младший разряд, а в последнем – старший. То есть на печать массив нужно выводить наоборот, чтобы число отображалось правильно.*  *Пример 2 демонстрирует сравнение двух больших чисел. Это необходимо, например, при реализации операции вычитания для двух больших чисел. Генерация большого числа реализована через функцию*  **Сложение и вычитание**  Базовые алгоритмы сложения, вычитания, умножения и деления чисел – хорошо известные алгоритмы выполнения действия «в столбик». Вместе с тем вся машинная арифметика сводится к одной операции сложения двух двоичных чисел. В этой связи, отдельного рассмотрения заслуживают задачи ускорения операции сложения. Основная идея ускорения состоит в раздельном рассмотрении поразрядного сложения и переносов. В двоичной системе это реализуется следующим образом, см. рисунок ниже, пример |



***Рисунок 1 – Способ ускорения операции сложения***

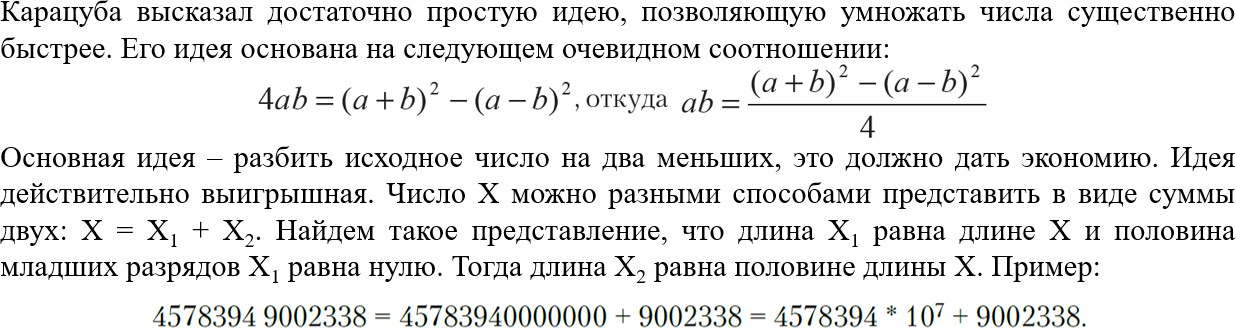
Здесь в качестве промежуточных значений используются два числа, промежуточная сумма и число переносов. Процесс завершается тогда, когда число переносов полностью обнуляется. На первом шаге суммируются исходные числа, на каждом последующем – промежуточная сумма и число переносов предыдущего шага. Промежуточная сумма формируется согласно следующему правилу: *k-ый разряд суммы равен 0, если равна 0 или 2 сумма k-ых разрядов слагаемых, k-ый разряд суммы равен 1, если равна 1 сумма k-ых разрядов слагаемых.*

Число переносов формируется согласно следующему правилу: *самый младший разряд числа всегда равен 0. k-ый разряд равен 1, если сумма (k–1)-ых разрядов равна 2. В иных случаях разряд числа переносов равен 0.*

# Умножение

Умножение столбиком можно назвать естественным алгоритмом, в том смысле, что столбиковое умножение непосредственно опирается на таблицу умножения цифр и определение операции умножения через сложение. Естественные идеи такого рода хороши для реализации, так как предполагают только некоторые технические сложности и не более того, но вряд ли так можно получить максимум эффективности.

# Метод Карацубы



При этом если числа являются разной длины или имеют нечетную длину, то целесообразно добавить старший «нулевой разряд». Например, если даны числа 37656 и 6567863, то их следует рассматривать в таком виде:

00037656 = 0003\*104 + 7656;

06567863 = 0656\*104 + 7863;

Будем рассматривать произведение двух чисел в виде (ax+b)(cx+d), где x=10k (в примере выше k=4). Имеет место следующая цепочка равенств

Для наших двух чисел имеем:

(0003\*104 + 7656)\* (0656\*104 + 7863)=

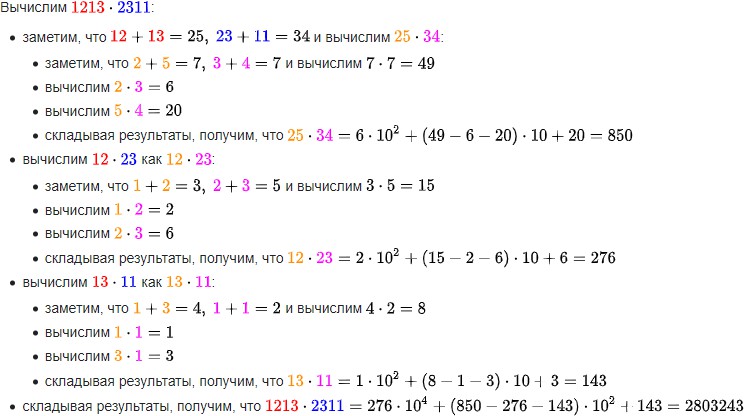
0003\*0656\*108+((0003+7656)\*(0656+7863)-0003\*0656-7656\*7863)\*104+7656\*7863

В этом выражении произведения 0003\*0656 и 7656\*7863 повторяются дважды, а всего нужно посчитать 3 произведения четырехразрядных чисел:

1. 0003\*0656
2. 7656\*7863
3. (0003+7656)\*(0656+7863) = 7659\*8519

Каждое произведение четырехразрядных чисел можно разбить на произведения двухразрядных, а произведения двухразрядных – на произведения одноразрядных чисел.

Пример для умножения четырехразрядных чисел методом Карацубы:



# Деление. Алгоритм с использованием метода половинного деления

Пусть некоторое время нас интересует только частное. Частное можно представить как неизвестную величину в уравнении вида:

Как известно, такие уравнения можно решать методом половинного деления. Здесь *f = A–Bp*, левая граница отрезка, содержащего корень, есть 0 и правая граница равна делимому A. Решением уравнения будет частное *p.* Далее остаток находится просто. Если частное получено с точностью до целого, то остаток легко определяется из формулы *A = Bp + q* ⇔ *q = A − Bp*

Вместо метода половинного деления можно использовать и другие методы: метод хорд или касательных

|  |  |
| --- | --- |
| **Пример 1** | |
| ***Задача:*** | |
|  | Написать на языке C++ программу, в которой создается динамический  массив для хранения длинного целого числа (более 10 цифр). Размер массива задается с клавиатуры. Цифры генерируются случайным образом. |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |

|  |  |
| --- | --- |
| **Пример 2** | |
| ***Задача:*** | |
|  | *Написать на языке C++ программу для сравнения двух больших чисел*  *(более 10 цифр)* |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Задание 1** | |
| ***Задача:*** | |
|  | Написать программу для сложения и вычитания двух больших чисел в столбик. Для представления чисел использовать массив, в котором будут храниться цифры числа. При реализации процедуры вычитания предпочтительно из большего вычитать меньшее. Для этого  первоначально определять, какое из двух чисел больше |
| ***Решение:*** | |
|  | #include <iostream> #include <string>  int\* create\_number(std::string s) {  int\* num = new int[s.length()];  or (int i = 0; i < s.length(); i++) {  char ch = s[i];  num[i] = ch - '0';  }  return num; }  void align(int\* array1, int len1, int\* array2, int len2) {  if (len1 > len2) {  for (int i = 0; i < len2; i++) {  array2[i+len1-len2] = array2[i];  }  for (int i = 0; i < len1-len2; i++) {  array2[i] = 0;  }  }  else {  for (int i = 0; i < len1; i++) {  array1[i+len2-len1] = array1[i];  }  for (int i = 0; i < len2-len1; i++) {  array1[i] = 0;  }  } }  bool compare(int\* num1, int\* num2, int len) {  for (int i = 0; i < len; i++) {  if (num1[i] < num2[i]) {  return false;  }  else if (num1[i] > num2[i]) {  return true;  }  }  return false; }  void plus(int\* array1, int\* array2, int len, int\* x) {  for (int i = 0; i < len; i++) {  x[i] = 0;  }  int p = 0;  for (int i = len-1; i >= 0; i--) {  if (i - 1 >= 0) {  x[i] = (p + array1[i-1] + array2[i-1]) % 10;  p = (p + array1[i-1] + array2[i-1] ) / 10;  }  else {  x[i] = p;  }  } }  void minus(int\* array1, int\* array2, int len, int\* x) {  for (int i = 0; i < len; i++) {  x[i] = 0;  }  for (int i = len; i >= 0; i--) {  if (array1[i] < array2[i]) {  int z = i - 1;  while (array1[z] == 0) {  z--;  }  array1[z] = array1[z] - 1;  z++;  while (array1[z] == 0 && z < i) {  array1[z] = 9;  z++;  }  x[i] = array1[i] - array2[i] + 10;  }  else {  x[i] = array1[i] - array2[i];  }  } }  int main() {  std::string s;  int num1\_len, num2\_len;  // Создание первого числа  std::cout << "Введите первое число: ";  std::cin >> s;  num1\_len = s.length(); // длина 1-го числа  int\* num1 = create\_number(s);    // Второго  std::cout << "Введите второе число: ";  std::cin >> s;  num2\_len = s.length(); // длина 2-го числа  int\* num2 = create\_number(s);    if (num1\_len != num2\_len) {  align(num1, num1\_len, num2, num2\_len);  if (num1\_len > num2\_len) {  num2\_len = num1\_len;  }  else {  num1\_len = num2\_len;  }  }    int res[num1\_len + 1];  plus(num1, num2, num1\_len + 1, res);      if (res[0] == 0) {  for (int i = 1; i < num1\_len + 1; i ++) {  std::cout << res[i];  }  }  else {  for (int i = 0; i < num1\_len + 1; i ++) {  std::cout << res[i];  }  }    std::cout << std::endl;    if (compare(num1,num2, num1\_len)) {  // 1-ое число больше 2-го  minus(num1, num2, num1\_len, res);  }  else {  minus(num2, num1, num1\_len, res);  }    for (int i = 0; i < num1\_len; i ++) {  std::cout << res[i];  }       delete[] num1;  delete[] num2;  return 0;  } |
| ***Ответ:*** | |
|  |  |
| **Задание 2\*** | |
| ***Задача:*** | |
|  | Написать программу для реализации ускоренного сложения чисел в  двоичной системе (аналогично примеру на рис. 1) |
| ***Решение:*** | |
|  | #include <iostream> #include <string> #include <numeric>  int\* create\_number(std::string s) {  int\* num = new int[s.length()];  for (int i = 0; i < s.length(); i++) {  char ch = s[i];  num[i] = ch - '0';  }  return num; }  void align(int\*& array1, int& len1, int\*& array2, int& len2) {  // Определяем новую длину для обоих массивов  int new\_len = std::max(len1, len2) + 1;   // Создаем новые массивы с увеличенной длиной  int\* new\_array1 = new int[new\_len];  int\* new\_array2 = new int[new\_len];   // заполняем новые массивы нулями  for (int i = 0; i < new\_len; i++) {  new\_array1[i] = 0;  new\_array2[i] = 0;  }   // Копируем старые значения в новые массивы с учетом смещения  for (int i = 0; i < len1; i++) {  new\_array1[new\_len - len1 + i] = array1[i];  }   for (int i = 0; i < len2; i++) {  new\_array2[new\_len - len2 + i] = array2[i];  }   // Освобождаем старые массивы  delete[] array1;  delete[] array2;   // Обновляем указатели на новые массивы и их длины  array1 = new\_array1;  array2 = new\_array2;  len1 = len2 = new\_len; }   void add(int\*& array1, int\*& array2, int len) {  int\* new\_array1 = new int[len];  int\* new\_array2 = new int[len];  new\_array2[len - 1] = 0;  for (int i = len - 1; i >= 0; i--) {  new\_array1[i] = (array1[i] + array2[i]) % 2;  if (i > 0) {  if (array1[i] + array2[i] == 2) {  new\_array2[i - 1] = 1;  }   else {  new\_array2[i - 1] = 0;  }  }  }    // Освобождаем старые массивы  delete[] array1;  delete[] array2;   // Обновляем указатели на новые массивы  array1 = new\_array1;  array2 = new\_array2; }   int main() {  std::string s;  int num1\_len, num2\_len;  // Создание первого числа  std::cout << "Введите первое число: ";  std::cin >> s;  num1\_len = s.length(); // длина 1-го числа  int\* num1 = create\_number(s);    // Второго  std::cout << "Введите второе число: ";  std::cin >> s;  num2\_len = s.length(); // длина 2-го числа  int\* num2 = create\_number(s);    // "Выравнивание чисел"  align(num1, num1\_len, num2, num2\_len);    for (int i = 0; i < num1\_len; i++) {  std::cout << num1[i] << " ";  }    std::cout<< std::endl;    for (int i = 0; i < num2\_len; i++) {  std::cout << num2[i] << " ";  }    std::cout<< std::endl;  std::cout<< std::endl;    int sum;// Сумма элементов второго числа (для проверки на окончание сложения)  for (int i = 0; i < num2\_len; i++) {  sum = sum + num2[i];  }    while (sum != 0) {  sum = 0;  add(num1, num2, num1\_len);    for (int i = 0; i < num1\_len; i++) {  std::cout << num1[i] << " ";  }  std::cout << std::endl;  for (int i = 0; i < num2\_len; i++) {  std::cout << num2[i] << " ";  }  std::cout << std::endl << std::endl;    for (int i = 0; i < num2\_len; i++) {  sum = sum + num2[i];  }    }         delete[] num1;  delete[] num2;  return 0;  } |
| ***Ответ:*** | |
|  |  |
| **Задание 3\*** | |
| ***Задача:*** | |
|  | В задании 2 была реализована идея ускорения суммирования двоичных  чисел. Реализуйте сложение десятичных чисел с использованием подобного механизма ускорения. |
| ***Решение:*** | |
|  | #include <iostream> #include <string> #include <numeric>  int\* create\_number(std::string s) {  int\* num = new int[s.length()];  for (int i = 0; i < s.length(); i++) {  char ch = s[i];  num[i] = ch - '0';  }  return num; }  void align(int\*& array1, int& len1, int\*& array2, int& len2) {  // Определяем новую длину для обоих массивов  int new\_len = std::max(len1, len2) + 1;   // Создаем новые массивы с увеличенной длиной  int\* new\_array1 = new int[new\_len];  int\* new\_array2 = new int[new\_len];   // заполняем новые массивы нулями  for (int i = 0; i < new\_len; i++) {  new\_array1[i] = 0;  new\_array2[i] = 0;  }   // Копируем старые значения в новые массивы с учетом смещения  for (int i = 0; i < len1; i++) {  new\_array1[new\_len - len1 + i] = array1[i];  }   for (int i = 0; i < len2; i++) {  new\_array2[new\_len - len2 + i] = array2[i];  }   // Освобождаем старые массивы  delete[] array1;  delete[] array2;   // Обновляем указатели на новые массивы и их длины  array1 = new\_array1;  array2 = new\_array2;  len1 = len2 = new\_len; }   void add(int\*& array1, int\*& array2, int len) {  int\* new\_array1 = new int[len];  int\* new\_array2 = new int[len];  new\_array2[len - 1] = 0;  for (int i = len - 1; i >= 0; i--) {  new\_array1[i] = (array1[i] + array2[i]) % 10;  if (i > 0) {  if (array1[i] + array2[i] > 9) {  new\_array2[i - 1] = 1;  }   else {  new\_array2[i - 1] = 0;  }  }  }    // Освобождаем старые массивы  delete[] array1;  delete[] array2;   // Обновляем указатели на новые массивы  array1 = new\_array1;  array2 = new\_array2; }   int main() {  std::string s;  int num1\_len, num2\_len;  // Создание первого числа  std::cout << "Введите первое число: ";  std::cin >> s;  num1\_len = s.length(); // длина 1-го числа  int\* num1 = create\_number(s);    // Второго  std::cout << "Введите второе число: ";  std::cin >> s;  num2\_len = s.length(); // длина 2-го числа  int\* num2 = create\_number(s);    // "Выравнивание чисел"  align(num1, num1\_len, num2, num2\_len);    for (int i = 0; i < num1\_len; i++) {  std::cout << num1[i] << " ";  }    std::cout<< std::endl;    for (int i = 0; i < num2\_len; i++) {  std::cout << num2[i] << " ";  }    std::cout<< std::endl;  std::cout<< std::endl;    int sum;// Сумма элементов второго числа (для проверки на окончание сложения)  for (int i = 0; i < num2\_len; i++) {  sum = sum + num2[i];  }    while (sum != 0) {  sum = 0;  add(num1, num2, num1\_len);    for (int i = 0; i < num1\_len; i++) {  std::cout << num1[i] << " ";  }  std::cout << std::endl;  for (int i = 0; i < num2\_len; i++) {  std::cout << num2[i] << " ";  }  std::cout << std::endl << std::endl;    for (int i = 0; i < num2\_len; i++) {  sum = sum + num2[i];  }    }         delete[] num1;  delete[] num2;  return 0;  } |
| ***Ответ:*** | |
|  |  |
| **Задание 4** | |
| ***Задача:*** | |
|  | Написать программу для умножения двух больших чисел столбиком. Обратите внимание на количество цифр в произведении: их скорее всего  будет больше, чем в исходных числах |
| ***Решение:*** | |
|  | #include <iostream> #include <string> #include <numeric> #include <cmath>  int\* create\_number(std::string s) {  int\* num = new int[s.length()];  for (int i = 0; i < s.length(); i++) {  char ch = s[i];  num[i] = ch - '0';  }  return num; }  void align(int\*& array1, int& len1, int\*& array2, int& len2) {  // Определяем новую длину для обоих массивов  int new\_len = std::max(len1, len2) \* 2;   // Создаем новые массивы с увеличенной длиной  int\* new\_array1 = new int[new\_len];  int\* new\_array2 = new int[new\_len];   // заполняем новые массивы нулями  for (int i = 0; i < new\_len; i++) {  new\_array1[i] = 0;  new\_array2[i] = 0;  }   // Копируем старые значения в новые массивы с учетом смещения  for (int i = 0; i < len1; i++) {  new\_array1[new\_len - len1 + i] = array1[i];  }   for (int i = 0; i < len2; i++) {  new\_array2[new\_len - len2 + i] = array2[i];  }   // Освобождаем старые массивы  delete[] array1;  delete[] array2;   // Обновляем указатели на новые массивы и их длины  array1 = new\_array1;  array2 = new\_array2;  len1 = len2 = new\_len; }   long int add(int\*& array1, int\*& array2, int len) {  int s = 0;  for (int i = len - 1; i >= 0; i--) {    int inter[len - 1];  for (int j = 0; j < len; j++) {  inter[j] = 0;  }    int k = array2[i];  int p = 0;  for (int j = len - 1; j >=0; j--) {  inter[j] = (p + array1[j]\*k) % 10;  p = (p + array1[j]\*k) / 10;  }    for (int j = len - 1; j >= 0; j--) {  s = s + inter[j]\*pow(10, len - 1 - j + (len - 1 - i));  }    }  return s; }   int main() {  std::string s;  int num1\_len, num2\_len;  // Создание первого числа  std::cout << "Введите первое число: ";  std::cin >> s;  num1\_len = s.length(); // длина 1-го числа  int\* num1 = create\_number(s);    // Второго  std::cout << "Введите второе число: ";  std::cin >> s;  num2\_len = s.length(); // длина 2-го числа  int\* num2 = create\_number(s);    // "Выравнивание чисел"  align(num1, num1\_len, num2, num2\_len);    std:: cout << add(num1,num2, num1\_len);     delete[] num1;  delete[] num2;  return 0;  } |
| ***Ответ:*** | |
|  |  |

|  |  |
| --- | --- |
|  |  |
| **Задание 5\*** | |
| ***Задача:*** | |
|  | Написать программу для умножения двух больших чисел методом Карацубы. Код программы объединить с кодом для задания 4, чтобы иметь возможность сравнить результат умножения одних и тех же чисел  столбиком и методом Карацубы |
| ***Решение:*** | |
|  | #include <iostream> #include <string> #include <cmath>  std::string remove(const std::string& num) {  std::string str;  bool flag = 0;  for (int i = 0; i < num.length(); i++) {  if (flag == 1) {  str = str + num[i];  }  if (num[i] != '0' && flag == 0) {  str = str + num[i];  flag = 1;  }  if (flag == 0 && i == num.length() - 1) {  return "0";  }  }  return str; }  void align(std::string& array1, int& len1, std::string& array2, int& len2) {  // Определяем новую длину для обоих массивов  int new\_len = std::max(len1, len2) + 1;   while (array1.length() < new\_len) {  array1 = '0' + array1;  }  while (array2.length() < new\_len) {  array2 = '0' + array2;  }   len1 = len2 = new\_len; }   void add(std::string& array1, std::string& array2) {  int len = array1.length();  std::string new\_array1(len, '0');  std::string new\_array2(len, '0');  new\_array2[len - 1] = '0';  for (int i = len - 1; i >= 0; i--) {  int sum = (array1[i] - '0') + (array2[i] - '0');  new\_array1[i] = (sum % 10) + '0';  if (i > 0) {  new\_array2[i - 1] = (sum / 10) + '0';  }  }  array1 = new\_array1;  array2 = new\_array2; }   std::string plus(const std::string& num1, const std::string& num2) {  std::string num1\_copy = num1;  std::string num2\_copy = num2;  int num1\_len = num1\_copy.length();  int num2\_len = num2\_copy.length();    // "Выравнивание чисел"  align(num1\_copy, num1\_len, num2\_copy, num2\_len);   int sum = 0;  for (int i = 0; i < num2\_len; i++) {  sum += (num2\_copy[i] - '0');  }   while (sum != 0) {  sum = 0;  add(num1\_copy, num2\_copy);  for (int i = 0; i < num2\_len; i++) {  sum += (num2\_copy[i] - '0');  }  }  return remove(num1\_copy); }  std::string minus(const std::string& num1, const std::string& num2) {  std::string num1\_copy = num1;  std::string num2\_copy = num2;  int num1\_len = num1.length();  int num2\_len = num2.length();   align(num1\_copy, num1\_len, num2\_copy, num2\_len);   std::string res(num1\_len, '0');   for (int i = num1\_len - 1; i >= 0; i--) {  int f = num1\_copy[i] - '0';  int s = num2\_copy[i] - '0';  if (f < s) {  int z = i - 1;  while (z >= 0 && num1\_copy[z] == '0') {  num1\_copy[z] = '9';  z--;  }  if (z >= 0) {  num1\_copy[z] = (num1\_copy[z] - '0' - 1) + '0';  }  f += 10;  }  res[i] = (f - s) + '0';  }  return remove(res); }  void split(std::string s, std::string& A, std::string& B, std::string& X, int c) {  if (c != -1) {  while (s.length() != c) {  s = '0' + s;  }  }  if (s.length() == 1) {  A = s;  B = "0";  X = "1";  }  else {  X = "1";  A = s.substr(0, s.length() / 2);  B = remove(s.substr(s.length() / 2));  for (int i = A.length(); i < s.length(); i++) {  X = X + '0';  }  }    if (A.length() != B.length()) {  if (A.length() > B.length()) {  B = '0' + B;  }  else {  A = '0' + A;  }  }    if (c != -1) {  while (A.length() < c/2) {  A = '0' + A;  }  while (B.length() < c/2) {  B = '0' + B;  }  } }  std::string mult(const std::string& num1, const std::string& num2) {  int len1 = num1.length();  int len2 = num2.length();  if (len1 == 1 && len2 == 1) {  return std::to\_string(std::stoi(num1) \* std::stoi(num2));  }  else {  std::string A, B, C, D, X;  if (len1 == len2) {  split(num1, A, B, X, -1);  split(num2, C, D, X, -1);  }  else if (len1 > len2) {  if (len1 % 2 == 0) {  split(num1, A, B, X, -1);  split(num2, C, D, X, len1);  }  else {  split(num1, A, B, X, len1 + 1);  split(num2, C, D, X, len1 + 1);  }  }  else if (len1 < len2){  if (len2 % 2 == 0) {  split(num1, A, B, X, len2);  split(num2, C, D, X, -1);  }  else {  split(num1, A, B, X, len2 + 1);  split(num2, C, D, X, len2 + 1);  }  }  std::string AC = mult(A,C);  std::string AC\_X2 = AC;  std::string BD = mult(B,D);  std::string ABCD = mult(plus(A,B), plus(C,D));  for (int i = 1; i < X.length(); i++) {  AC\_X2 = AC\_X2 + '0' + '0';  }  std::string temp = minus(minus(ABCD, AC), BD);  for (int i = 1; i < X.length(); i++) {  temp = temp + '0';  }  return plus(plus(AC\_X2,temp), BD);  } }   int main() {  std::string num1 = "3765411241", A,B,C,D,X1,X2, num2 = "65642114";  std::cout << mult(num1,num2);     return 0; } |
| ***Ответ:*** | |
|  |  |
| **Задание 6** | |
| ***Задача:*** | |
|  | Написать программу для нахождения частного и остатка от деления двух  чисел методом половинного деления |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |