Задание 1

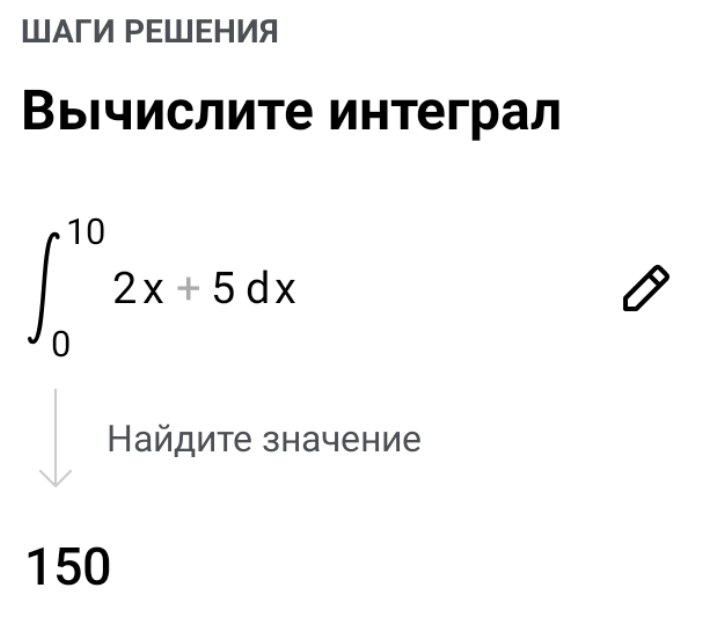
Когда я написал тот же код, что и в примере мне показало ошибки 

До стандарта C++98/03 преобразование из const char\* в char\* могло выполняться неявно, даже если это нарушало семантику const. Это означало, что вы могли случайно изменить данные, которые не должны быть изменяемыми, например строковые литералы. В C++11 и более поздних версиях, такое преобразование стало ошибкой компиляции, чтобы предотвратить возможные ошибки. Поэтому я немного изменил код программы (изменил типы параметров f1, f2, и callfun с char\* на const char\*)

|  |
| --- |
| #include <iostream>  using namespace std;  int f1(const char\* s) {  cout << s << " \nВыполняется тест №1" << endl;  return 1;  }  int f2(const char\* s) {  cout << s << " \nВыполняется тест №2" << endl;  return 2;  }  void callfun(const char\* s, int (\*f)(const char\*)) {  cout << " f() = " << f(s) << '\n';  }  int main() {  setlocale(0, "Rus");  callfun("Тест №1 вызван", f1);  callfun("Тест №2 вызван", f2);  } |

Задание 2

Была опять ошибка в функции main мы вызываем функцию metodPrjam, но её имя в коде определено как integ в прототипе. Я исправил имя в прототипе на metodPrjam. Также была ошибка в самом алгоритме реализации формулы правых прямоугольников. Тут возвращается результат с делением на 2: return s / 2. Для метода правых прямоугольников этого не нужно делать. В формуле правых прямоугольников используется просто сумма всех прямоугольников. Делить на 2 нужно в методе трапеций, где усредняется результат, полученный по левой и правой границам. Вот статьи, на которые я основывался ([ссылка\_1](https://ru.stackoverflow.com/questions/381771/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE-%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D0%B0-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BC-%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2), [ссылка\_2](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D1%83%D0%B3%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2?utm_source=chatgpt.com) ) + решил по определенному интегралу функцию и получается ответ 150. А в формуле из методички получалось 75. 



Исправленный вариант( ну мне кажется, что я прав)

|  |
| --- |
| #include <iostream>  float metodPrjam(float(\*) (float), float, float, float); // прототип  float f(float); // прототип  int main()  {  float z;  z = metodPrjam(f, (float)0.0, (float)10.0, (float)0.01);  std::cout << "Result=" << z;  }  float metodPrjam(float(\*fun) (float), float a, float b, float h)  {  float x, s = 0.0f;  x = a + h;  while (x <= b)  {  s += h \* fun(x);  x += h;  }  return s;  }  float f(float x)  {  return (2 \* x + 5);  } |

Задание 3

Условие задачи:

Написать функцию, которая принимает как минимум один обязательный параметр и произвольное количество дополнительных параметров, и выполняет над ними суммирование.

Ошибка:

Код программы, который дан в методичке содержит ошибку. Этот код полагается на то, что аргументы в памяти располагаются последовательно, что не гарантируется стандартом C++. Использование адресной арифметики для доступа к переменным аргументам является, такой подход может привести к ошибкам и непредсказуемому поведению программы как в моём случае. Вот что выводит программа



А это явно не то, что должно быть. Чтобы это исправить , я использовал [ссылка](https://cplusplus.com/reference/cstdarg/va_list/) для того, чтобы я мог обращаться к элементам последовательно

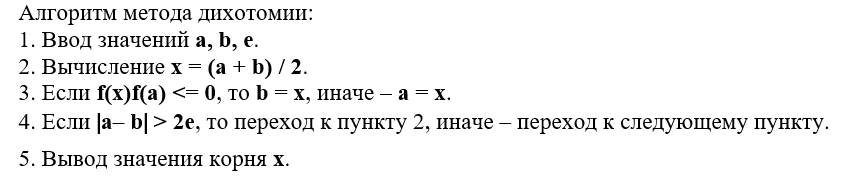
Исправленный код

|  |
| --- |
| #include <iostream>  #include <cstdarg>  int sum(int n, ...);  using namespace std;  int main() {  cout << sum(6, 4, 5, 1, 2, 3, 0) << endl;  cout << sum(2, 34, 4445) << endl;  return 0;  }  int sum(int n, ...) {  va\_list args;  va\_start(args, n);  int sum = 0;  for (int i = 0; i < n; i++) {  sum += va\_arg(args, int);  }  va\_end(args);  return sum;  } |

Задание 4

Вариант 12

|  |
| --- |
| 2x + x3 – 7,  e x + 2x |

Для нахождения по методу дихотомии я использовал алгоритм из лаб.№9 

|  |
| --- |
| #include <iostream>  #include <cmath>  using namespace std;  float equation1(float x) {  return 2 \* x + pow(x, 3) - 7;  }  float equation2(float x) {  return exp(x)+2\*x;  }  float dichotomy(float(\*f) (float), float a, float b, float e) {  float x;  do  {  x = (a + b) / 2;  if ((f(x) \* f(a)) <= 0)  b = x;  else  a = x;  } while (abs(a - b) >= 2 \* e);  return x;  }  int main() {  setlocale(LC\_ALL, "ru");  float a, b, e = 0.001;  float root1, root2;  cout << "Введите границы отрезка для первого уравнения: ";  cin >> a >> b;  float a2, b2;  cout << "Введите границы отрезка для второго уравнения : ";  cin >> a2 >> b2;  root1 = dichotomy(equation1, a, b, e);  cout << "Корень первого уравнения: " << root1 << endl;  root2 = dichotomy(equation2, a2, b2, e);  cout << "Корень второго уравнения: " << root2 << endl;  return 0;  } |

|  |
| --- |
| Написать функцию, проверяющую есть ли отрицательные элементы в заданном одномерном массиве размерностью **n**. Удалить из массива все отрицательные элементы, удаленный элемент заполняется нулем и переносится в конец массива. |

|  |
| --- |
| #include <iostream>  #include <cstdarg>  using namespace std;  void processArray(int n, ...) {  va\_list args;  va\_start(args, n);  int\* arr = new int[n];  for (int i = 0; i < n; i++) {  arr[i] = va\_arg(args, int);  }  va\_end(args);  int index = 0;  for (int i = 0; i < n; i++) {  if (arr[i] >= 0) {  arr[index++] = arr[i];  }  }  while (index < n) {  arr[index++] = 0;  }  cout << "Массив после изменения: ";  for (int i = 0; i < n; i++) {  cout << arr[i] << " ";  }  cout << endl;  delete[] arr;  }  int main() {  setlocale(LC\_ALL, "ru");  cout << "Пример 1:" << endl;  processArray(6, 1, -3, 4, -2, -5, 6);  cout << "Пример 2:" << endl;  processArray(5, 10, 20, 30, 40, 50);  cout << "Пример 3:" << endl;  processArray(4, -1, -2, -3, -4);  cout << "Пример 4:" << endl;  processArray(5, -10, 10, 9, -8, -11);  return 0;  } |

Доп.задания

Вариант 4 (12+8=20=4)

|  |
| --- |
| x3 + 2x – 1,  ex − 2 |

|  |
| --- |
| #include <iostream>  #include <cmath>  using namespace std;  float equation1(float x) {  return pow(x, 3)+ 2 \* x - 1;  }  float equation2(float x) {  return exp(x)-2;  }  float dichotomy(float(\*f) (float), float a, float b, float e) {  float x;  do  {  x = (a + b) / 2;  if ((f(x) \* f(a)) <= 0)  b = x;  else  a = x;  } while (abs(a - b) >= 2 \* e);  return x;  }  int main() {  setlocale(LC\_ALL, "ru");  float a, b, e = 0.001;  float root1, root2;  cout << "Введите границы отрезка для первого уравнения: ";  cin >> a >> b;  float a2, b2;  cout << "Введите границы отрезка для второго уравнения : ";  cin >> a2 >> b2;  root1 = dichotomy(equation1, a, b, e);  cout << "Корень первого уравнения: " << root1 << endl;  root2 = dichotomy(equation2, a2, b2, e);  cout << "Корень второго уравнения: " << root2 << endl;  return 0;  } |

|  |
| --- |
| Написать функцию **days** с переменным числом параметров, которая находит количество дней, прошедших между двумя датами (параметрами функции являются даты в формате «дд.мм.гг». |

|  |
| --- |
| #include <iostream>  #include <cstdarg>  using namespace std;  bool isLeapYear(int year) {  return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);  }  int daysInMonth(int month, int year) {  int days[] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };  if (month == 2 && isLeapYear(year)) {  return 29;  }  return days[month];  }  void parseDate(const char\* dateStr, int& day, int& month, int& year) {  day = (dateStr[0] - '0') \* 10 + (dateStr[1] - '0');  month = (dateStr[3] - '0') \* 10 + (dateStr[4] - '0');  year = (dateStr[6] - '0') \* 1000 + (dateStr[7] - '0') \* 100 + (dateStr[8] - '0') \* 10 + (dateStr[9] - '0');  }  int daysBetween(int d1, int m1, int y1, int d2, int m2, int y2) {  int days = 0;  while (y1 < y2 || m1 < m2 || d1 < d2)  {  d1++;  if (d1 > daysInMonth(m1, y1))  {  d1 = 1;  m1++;  if (m1 > 12)  {  m1 = 1;  y1++;  }  }  days++;  }  return days;  }  void days(int numPairs, ...) {  va\_list args;  va\_start(args, numPairs);  for (int i = 0; i < numPairs; ++i) {  const char\* dateStr1 = va\_arg(args, const char\*);  const char\* dateStr2 = va\_arg(args, const char\*);  int d1, m1, y1, d2, m2, y2;  parseDate(dateStr1, d1, m1, y1);  parseDate(dateStr2, d2, m2, y2);  int diff = daysBetween(d1, m1, y1, d2, m2, y2);  cout << "Разница между " << dateStr1 << " и " << dateStr2 << ": " << diff << " дней" << endl;  }  va\_end(args);  }  int main() {  setlocale(LC\_ALL, "ru");  days(3, "01.01.2023", "10.01.2023", "15.03.2024", "20.04.2024", "01.05.2022", "01.01.2023");  days(1, "01.01.2000", "01.01.2001");  days(2, "10.10.2020", "20.10.2020", "01.01.1990", "01.01.2000");  return 0;  } |

Вариант 16 (12+4=16)

|  |
| --- |
| sin(x) + 2 + x,  2 + x3 |

|  |
| --- |
| #include <iostream>  #include <cmath>  using namespace std;  float equation1(float x) {  return sin(x)+2+x;  }  float equation2(float x) {  return 2+pow(x,3);  }  float dichotomy(float(\*f) (float), float a, float b, float e) {  float x;  do  {  x = (a + b) / 2;  if ((f(x) \* f(a)) <= 0)  b = x;  else  a = x;  } while (abs(a - b) >= 2 \* e);  return x;  }  int main() {  setlocale(LC\_ALL, "ru");  float a, b, e = 0.001;  float root1, root2;  cout << "Введите границы отрезка для первого уравнения: ";  cin >> a >> b;  float a2, b2;  cout << "Введите границы отрезка для второго уравнения : ";  cin >> a2 >> b2;  root1 = dichotomy(equation1, a, b, e);  cout << "Корень первого уравнения: " << root1 << endl;  root2 = dichotomy(equation2, a2, b2, e);  cout << "Корень второго уравнения: " << root2 << endl;  return 0;  } |

|  |
| --- |
| Написать функцию, которая определяет, есть ли в тексте слова, начинающиеся с буквы **а**, и сколько таких слов. С помощью этой функции проверить несколько строк.  (я сделал, что он проверяет на букву а(русскую) в любом регистре) |

|  |
| --- |
| #include <iostream>  #include <vector>  #include <string>  #include <cstdarg>  #include <cctype>  using namespace std;  char tolower\_ru(char ch) {  if (ch >= 'А' && ch <= 'Я') {  return ch + ('а' - 'А');  }  return ch;  }  int countWordsStartingWithA(const string& sentence) {  vector<string> words;  size\_t start = 0, end;  // разбиваем строки на слова  while ((end = sentence.find(' ', start)) != string::npos) {  words.push\_back(sentence.substr(start, end - start));  start = end + 1;  }  words.push\_back(sentence.substr(start));  int count = 0;  for (const string& word : words) {  if (!word.empty() && tolower\_ru(word[0]) == 'а') {  count++;  }  }  return count;  }  void checkWordsWithA(int numStrings, ...) {  va\_list args;  va\_start(args, numStrings);  for (int i = 0; i < numStrings; ++i) {  const char\* text = va\_arg(args, const char\*);  string textStr(text);  if (textStr.empty()) {  cout << "Пустая строка!" << endl;  continue;  }  int result = countWordsStartingWithA(textStr);  cout << "В строке \"" << text << "\" количество слов на 'а': " << result << endl;  }  va\_end(args);  }  int main() {  setlocale(LC\_ALL, "ru");  checkWordsWithA(3,  "Апельсин и арбуз — это фрукты.",  "Это обычный текст без слов на 'a'(тут латинская)",  "Академия и анализ — важные понятия.");  cout << endl;  checkWordsWithA(2,  "Автомобиль и автобус — это транспорт.",  "Тут есть слова на а автобус.");  cout << endl;  checkWordsWithA(1,  "Алиса и аркадия — два разных имени.");  return 0;  } |

Вариант 8 (12-4=8)

|  |
| --- |
| x3 + 3x – 1,  ex − 4 |

|  |
| --- |
| #include <iostream>  #include <cmath>  using namespace std;  float equation1(float x) {  return pow(x,3)+3\*x-1;  }  float equation2(float x) {  return exp(x)-4;  }  float dichotomy(float(\*f) (float), float a, float b, float e) {  float x;  do  {  x = (a + b) / 2;  if ((f(x) \* f(a)) <= 0)  b = x;  else  a = x;  } while (abs(a - b) >= 2 \* e);  return x;  }  int main() {  setlocale(LC\_ALL, "ru");  float a, b, e = 0.001;  float root1, root2;  cout << "Введите границы отрезка для первого уравнения: ";  cin >> a >> b;  float a2, b2;  cout << "Введите границы отрезка для второго уравнения : ";  cin >> a2 >> b2;  root1 = dichotomy(equation1, a, b, e);  cout << "Корень первого уравнения: " << root1 << endl;  root2 = dichotomy(equation2, a2, b2, e);  cout << "Корень второго уравнения: " << root2 << endl;  return 0;  } |

|  |
| --- |
| Написать функцию **prost** с переменным числом параметров, которая находит все простые числа из нескольких интервалов. Интервалы задаются границами **a** и **b**. С помощью этой функции проверить несколько интервалов. |

|  |
| --- |
| #include <iostream>  #include <cmath>  #include <cstdarg>  #include <vector>  using namespace std;  bool isPrime(int n) {  if (n <= 1) return false;  if (n == 2) return true;  if (n % 2 == 0) return false;  for (int i = 3; i <= sqrt(n); i += 2) {  if (n % i == 0) return false;  }  return true;  }  void prost(int numIntervals, ...) {  va\_list args;  va\_start(args, numIntervals);  for (int i = 0; i < numIntervals; ++i) {  int a = va\_arg(args, int);  int b = va\_arg(args, int);  cout << "Простые числа в интервале от " << a << " до " << b << ": ";  for (int j = a; j <= b; ++j) {  if (isPrime(j)) {  cout << j << " ";  }  }  cout << endl;  }  va\_end(args);  }  int main() {  setlocale(LC\_ALL, "ru");  prost(3, 10, 30, 50, 100, 200, 300);  return 0;  } |