Задача 3.1 Выдача в формате программы на plain "C"

Написать программу, которая находит явный вид двух интерполирующих полиномов для функции $f(x) = 1/\cosh(x - 1/100)$. Воспользуйтесь формулой Лагранжа

$$P(x) = \sum_{i=1}^{N} f(x_i) \prod_{k=1, k\neq i}^{N} \frac{(x - x_k)}{(x_i - x_k)}$$

Для первого полинома $N = 21$, $x_i = -1.6 + 0.16 * (i - 1)$ при $i = 1 \dots 21$. Для второго — $N = 21$, $x_i = -1.6 * \cos(\pi * (i - 1)/20)$ при $i = 1 \dots 21$.

После этого она должна создать файл, который является программой на языке "plain C", работающей под `gcc`. Ручная правка полученного файла не допускается. Программа должна определять 3 функции:

`double f(double x)` — исходная функция;

`double p1(double x)` — первый интерполирующий полином;

`double p2(double x)` — второй интерполирующий полином;

Она должна сравнивать результаты интерполяции и находить максимальное отклонение от функции для первого и второго полиномов на наборе точек $t_i = -1.6 + 0.016 * (i - 1)$ при $i = 1 \dots 201$. Выходной файл, который создает программа на "C", должен содержать

$$\Delta^{(1)}_{max} \qquad \Delta^{(2)}_{max}$$

и таблицу

$$t_i \qquad f(t_i) \qquad \Delta^{(1)}_i \qquad \Delta^{(2)}_i$$

где $\Delta^{(1)}_i = f(t_i) - P_1(t_i)$, $\Delta^{(2)}_i = f(t_i) - P_2(t_i)$.

Сообразите, отчего такая разница между $\Delta^{(1)}$ и $\Delta^{(2)}$.

Замечание про вывод в файл.

Функции вывода в файл в разных версиях MAXIM'ы работают по-разному. Например, в текущей версии функция writefile почему-то делает append, а не overwrite существующего файла. Регулярный и не зависящий от версии способ выводить в файл (кстати, вывод будет идти без излишней в данном случае служебной информации) — это функция `with_stdout` в сочетании с флагом `file_output_append`:

```
file_output_append:false;
with_stdout("myfile.out", print("line 1"));
file_output_append:true;
with_stdout("myfile.out", print("line 2"));
with_stdout("myfile.out", print("line 3"));
```

(здесь первые 2 строки гарантированно затирают предыдущее содержимое файла и выводят в файл "line 1", а следующие строки позволяют дописать "line2", "line3", и т.д.)

Печать двойной кавычки и обратного слеша производится обычным образом:

```
print("Double quote: \"    ");
print("Backslash: \\    ");
```

Не мучайте компилятор `gcc` излишне длинными выражениями ("x", умноженный на себя 20 раз — это ужасно, функция `pow(x,20)` — это безграмотно, и сумма ряда, записанная в явном виде, а не в виде цикла — это ужасно, присвоения значений коэффициентов ряда при каждом вызове функции — это безграмотно, отдельное вычисление x^{11}, притом что шаг назад был сосчитано x^{10} — это ужасно, и т.д.).

Задача 3.2 Выдача в формате программы на plain "C"

Написать программу, которая находит явный вид двух интерполирующих полиномов для функции $f(x) = 1/\cosh(x - 2/100)$. Воспользуйтесь формулой Лагранжа

$$P(x) = \sum_{i=1}^{N} f(x_i) \prod_{k=1,k\neq i}^{N} \frac{(x - x_k)}{(x_i - x_k)}$$

Для первого полинома $N = 21$, $x_i = -1.6 + 0.16 * (i - 1)$ при $i = 1 \dots 21$. Для второго — $N = 21$, $x_i = -1.6 * \cos(\pi * (i - 1)/20)$ при $i = 1 \dots 21$.

После этого она должна создать файл, который является программой на языке "plain C", работающей под `gcc`. Ручная правка полученного файла не допускается. Программа должна определять 3 функции:

`double f(double x)` — исходная функция;

`double p1(double x)` — первый интерполирующий полином;

`double p2(double x)` — второй интерполирующий полином;

Она должна сравнивать результаты интерполяции и находить максимальное отклонение от функции для первого и второго полиномов на наборе точек $t_i = -1.6 + 0.016 * (i - 1)$ при $i = 1 \dots 201$. Выходной файл, который создает программа на "C", должен содержать

$$\Delta^{(1)}_{max} \qquad \Delta^{(2)}_{max}$$

и таблицу

$$t_i \qquad f(t_i) \qquad \Delta^{(1)}_i \qquad \Delta^{(2)}_i$$

где $\Delta^{(1)}_i = f(t_i) - P_1(t_i)$, $\Delta^{(2)}_i = f(t_i) - P_2(t_i)$.

Сообразите, отчего такая разница между $\Delta^{(1)}$ и $\Delta^{(2)}$.

Замечание про вывод в файл.

Функции вывода в файл в разных версиях MAXIM'ы работают по-разному. Например, в текущей версии функция writefile почему-то делает append, а не overwrite существующего файла. Регулярный и не зависящий от версии способ выводить в файл (кстати, вывод будет идти без излишней в данном случае служебной информации) — это функция `with_stdout` в сочетании с флагом `file_output_append`:

```
file_output_append:false;
with_stdout("myfile.out", print("line 1"));
file_output_append:true;
with_stdout("myfile.out", print("line 2"));
with_stdout("myfile.out", print("line 3"));
```

(здесь первые 2 строки гарантированно затирают предыдущее содержимое файла и выводят в файл "line 1", а следующие строки позволяют дописать "line2", "line3", и т.д.)

Печать двойной кавычки и обратного слеша производится обычным образом:

```
print("Double quote: \"    ");
print("Backslash: \\    ");
```

Не мучайте компилятор `gcc` излишне длинными выражениями ("x", умноженный на себя 20 раз — это ужасно, функция `pow(x,20)` — это безграмотно, и сумма ряда, записанная в явном виде, а не в виде цикла — это ужасно, присвоения значений коэффициентов ряда при каждом вызове функции — это безграмотно, отдельное вычисление x^{11}, притом что шаг назад был сосчитано x^{10} — это ужасно, и т.д.).

Задача 3.3 Выдача в формате программы на plain "C"

Написать программу, которая находит явный вид двух интерполирующих полиномов для функции $f(x) = 1/\cosh(x - 3/100)$. Воспользуйтесь формулой Лагранжа

$$P(x) = \sum_{i=1}^{N} f(x_i) \prod_{k=1, k\neq i}^{N} \frac{(x - x_k)}{(x_i - x_k)}$$

Для первого полинома $N = 21$, $x_i = -1.6 + 0.16 * (i - 1)$ при $i = 1 \dots 21$. Для второго — $N = 21$, $x_i = -1.6 * \cos(\pi * (i - 1)/20)$ при $i = 1 \dots 21$.

После этого она должна создать файл, который является программой на языке "plain C", работающей под `gcc`. Ручная правка полученного файла не допускается. Программа должна определять 3 функции:

`double f(double x)` — исходная функция;

`double p1(double x)` — первый интерполирующий полином;

`double p2(double x)` — второй интерполирующий полином;

Она должна сравнивать результаты интерполяции и находить максимальное отклонение от функции для первого и второго полиномов на наборе точек $t_i = -1.6 + 0.016 * (i - 1)$ при $i = 1 \dots 201$. Выходной файл, который создает программа на "C", должен содержать

$$\Delta^{(1)}_{max} \qquad \Delta^{(2)}_{max}$$

и таблицу

$$t_i \qquad f(t_i) \qquad \Delta^{(1)}_i \qquad \Delta^{(2)}_i$$

где $\Delta^{(1)}_i = f(t_i) - P_1(t_i)$, $\Delta^{(2)}_i = f(t_i) - P_2(t_i)$.

Сообразите, отчего такая разница между $\Delta^{(1)}$ и $\Delta^{(2)}$.

Замечание про вывод в файл.

Функции вывода в файл в разных версиях MAXIM'ы работают по-разному. Например, в текущей версии функция writefile почему-то делает append, а не overwrite существующего файла. Регулярный и не зависящий от версии способ выводить в файл (кстати, вывод будет идти без излишней в данном случае служебной информации) — это функция `with_stdout` в сочетании с флагом `file_output_append`:

```
file_output_append:false;
with_stdout("myfile.out", print("line 1"));
file_output_append:true;
with_stdout("myfile.out", print("line 2"));
with_stdout("myfile.out", print("line 3"));
```

(здесь первые 2 строки гарантированно затирают предыдущее содержимое файла и выводят в файл "line 1", а следующие строки позволяют дописать "line2", "line3", и т.д.)

Печать двойной кавычки и обратного слеша производится обычным образом:

```
print("Double quote: \"    ");
print("Backslash: \\    ");
```

Не мучайте компилятор `gcc` излишне длинными выражениями ("x", умноженный на себя 20 раз — это ужасно, функция `pow(x,20)` — это безграмотно, и сумма ряда, записанная в явном виде, а не в виде цикла — это ужасно, присвоения значений коэффициентов ряда при каждом вызове функции — это безграмотно, отдельное вычисление x^{11}, притом что шаг назад был сосчитано x^{10} — это ужасно, и т.д.).

Задача 3.4 Выдача в формате программы на plain "C"

Написать программу, которая находит явный вид двух интерполирующих полиномов для функции $f(x) = 1/\cosh(x - 4/100)$. Воспользуйтесь формулой Лагранжа

$$P(x) = \sum_{i=1}^{N} f(x_i) \prod_{k=1,k\neq i}^{N} \frac{(x - x_k)}{(x_i - x_k)}$$

Для первого полинома $N = 21$, $x_i = -1.6 + 0.16 * (i - 1)$ при $i = 1 \dots 21$. Для второго — $N = 21$, $x_i = -1.6 * \cos(\pi * (i - 1)/20)$ при $i = 1 \dots 21$.

После этого она должна создать файл, который является программой на языке "plain C", работающей под `gcc`. Ручная правка полученного файла не допускается. Программа должна определять 3 функции:

`double f(double x)` — исходная функция;

`double p1(double x)` — первый интерполирующий полином;

`double p2(double x)` — второй интерполирующий полином;

Она должна сравнивать результаты интерполяции и находить максимальное отклонение от функции для первого и второго полиномов на наборе точек $t_i = -1.6 + 0.016 * (i - 1)$ при $i = 1 \dots 201$. Выходной файл, который создает программа на "C", должен содержать

$$\Delta^{(1)}_{max} \qquad \Delta^{(2)}_{max}$$

и таблицу

$$t_i \qquad f(t_i) \qquad \Delta^{(1)}_i \qquad \Delta^{(2)}_i$$

где $\Delta^{(1)}_i = f(t_i) - P_1(t_i)$, $\Delta^{(2)}_i = f(t_i) - P_2(t_i)$.

Сообразите, отчего такая разница между $\Delta^{(1)}$ и $\Delta^{(2)}$.

Замечание про вывод в файл.

Функции вывода в файл в разных версиях MAXIM'ы работают по-разному. Например, в текущей версии функция writefile почему-то делает append, а не overwrite существующего файла. Регулярный и не зависящий от версии способ выводить в файл (кстати, вывод будет идти без излишней в данном случае служебной информации) — это функция `with_stdout` в сочетании с флагом `file_output_append`:

```
file_output_append:false;
with_stdout("myfile.out", print("line 1"));
file_output_append:true;
with_stdout("myfile.out", print("line 2"));
with_stdout("myfile.out", print("line 3"));
```

(здесь первые 2 строки гарантированно затирают предыдущее содержимое файла и выводят в файл "line 1", а следующие строки позволяют дописать "line2", "line3", и т.д.)

Печать двойной кавычки и обратного слеша производится обычным образом:

```
print("Double quote: \"    ");
print("Backslash: \\    ");
```

Не мучайте компилятор `gcc` излишне длинными выражениями ("x", умноженный на себя 20 раз — это ужасно, функция `pow(x,20)` — это безграмотно, и сумма ряда, записанная в явном виде, а не в виде цикла — это ужасно, присвоения значений коэффициентов ряда при каждом вызове функции — это безграмотно, отдельное вычисление x^{11}, притом что шаг назад был сосчитано x^{10} — это ужасно, и т.д.).

Задача 3.5 Выдача в формате программы на plain "C"

Написать программу, которая находит явный вид двух интерполирующих полиномов для функции $f(x) = 1/\cosh(x - 5/100)$. Воспользуйтесь формулой Лагранжа

$$P(x) = \sum_{i=1}^{N} f(x_i) \prod_{k=1, k\neq i}^{N} \frac{(x - x_k)}{(x_i - x_k)}$$

Для первого полинома $N = 21$, $x_i = -1.6 + 0.16 * (i - 1)$ при $i = 1 \dots 21$. Для второго — $N = 21$, $x_i = -1.6 * \cos(\pi * (i - 1)/20)$ при $i = 1 \dots 21$.

После этого она должна создать файл, который является программой на языке "plain C", работающей под `gcc`. Ручная правка полученного файла не допускается. Программа должна определять 3 функции:

`double f(double x)` — исходная функция;

`double p1(double x)` — первый интерполирующий полином;

`double p2(double x)` — второй интерполирующий полином;

Она должна сравнивать результаты интерполяции и находить максимальное отклонение от функции для первого и второго полиномов на наборе точек $t_i = -1.6 + 0.016 * (i - 1)$ при $i = 1 \dots 201$. Выходной файл, который создает программа на "C", должен содержать

$$\Delta^{(1)}_{max} \qquad \Delta^{(2)}_{max}$$

и таблицу

$$t_i \qquad f(t_i) \qquad \Delta^{(1)}_i \qquad \Delta^{(2)}_i$$

где $\Delta^{(1)}_i = f(t_i) - P_1(t_i)$, $\Delta^{(2)}_i = f(t_i) - P_2(t_i)$.

Сообразите, отчего такая разница между $\Delta^{(1)}$ и $\Delta^{(2)}$.

Замечание про вывод в файл.

Функции вывода в файл в разных версиях MAXIM'ы работают по-разному. Например, в текущей версии функция writefile почему-то делает append, а не overwrite существующего файла. Регулярный и не зависящий от версии способ выводить в файл (кстати, вывод будет идти без излишней в данном случае служебной информации) — это функция `with_stdout` в сочетании с флагом `file_output_append`:

```
file_output_append:false;
with_stdout("myfile.out", print("line 1"));
file_output_append:true;
with_stdout("myfile.out", print("line 2"));
with_stdout("myfile.out", print("line 3"));
```

(здесь первые 2 строки гарантированно затирают предыдущее содержимое файла и выводят в файл "line 1", а следующие строки позволяют дописать "line2", "line3", и т.д.)

Печать двойной кавычки и обратного слеша производится обычным образом:

```
print("Double quote: \"    ");
print("Backslash: \\    ");
```

Не мучайте компилятор `gcc` излишне длинными выражениями ("x", умноженный на себя 20 раз — это ужасно, функция `pow(x,20)` — это безграмотно, и сумма ряда, записанная в явном виде, а не в виде цикла — это ужасно, присвоения значений коэффициентов ряда при каждом вызове функции — это безграмотно, отдельное вычисление x^{11}, притом что шаг назад был сосчитано x^{10} — это ужасно, и т.д.).

Задача 3.6 Выдача в формате программы на plain "C"

Написать программу, которая находит явный вид двух интерполирующих полиномов для функции $f(x) = 1/\cosh(x - 6/100)$. Воспользуйтесь формулой Лагранжа

$$P(x) = \sum_{i=1}^{N} f(x_i) \prod_{k=1,k\neq i}^{N} \frac{(x - x_k)}{(x_i - x_k)}$$

Для первого полинома $N = 21$, $x_i = -1.6 + 0.16 * (i - 1)$ при $i = 1 \dots 21$. Для второго — $N = 21$, $x_i = -1.6 * \cos(\pi * (i - 1)/20)$ при $i = 1 \dots 21$.

После этого она должна создать файл, который является программой на языке "plain C", работающей под `gcc`. Ручная правка полученного файла не допускается. Программа должна определять 3 функции:

`double f(double x)` — исходная функция;

`double p1(double x)` — первый интерполирующий полином;

`double p2(double x)` — второй интерполирующий полином;

Она должна сравнивать результаты интерполяции и находить максимальное отклонение от функции для первого и второго полиномов на наборе точек $t_i = -1.6 + 0.016 * (i - 1)$ при $i = 1 \dots 201$. Выходной файл, который создает программа на "C", должен содержать

$$\Delta^{(1)}_{max} \qquad \Delta^{(2)}_{max}$$

и таблицу

$$t_i \qquad f(t_i) \qquad \Delta^{(1)}_i \qquad \Delta^{(2)}_i$$

где $\Delta^{(1)}_i = f(t_i) - P_1(t_i)$, $\Delta^{(2)}_i = f(t_i) - P_2(t_i)$.

Сообразите, отчего такая разница между $\Delta^{(1)}$ и $\Delta^{(2)}$.

Замечание про вывод в файл.

Функции вывода в файл в разных версиях MAXIM'ы работают по-разному. Например, в текущей версии функция writefile почему-то делает append, а не overwrite существующего файла. Регулярный и не зависящий от версии способ выводить в файл (кстати, вывод будет идти без излишней в данном случае служебной информации) — это функция `with_stdout` в сочетании с флагом `file_output_append`:

```
file_output_append:false;
with_stdout("myfile.out", print("line 1"));
file_output_append:true;
with_stdout("myfile.out", print("line 2"));
with_stdout("myfile.out", print("line 3"));
```

(здесь первые 2 строки гарантированно затирают предыдущее содержимое файла и выводят в файл "line 1", а следующие строки позволяют дописать "line2", "line3", и т.д.)

Печать двойной кавычки и обратного слеша производится обычным образом:

```
print("Double quote: \"    ");
print("Backslash: \\    ");
```

Не мучайте компилятор `gcc` излишне длинными выражениями ("x", умноженный на себя 20 раз — это ужасно, функция `pow(x,20)` — это безграмотно, и сумма ряда, записанная в явном виде, а не в виде цикла — это ужасно, присвоения значений коэффициентов ряда при каждом вызове функции — это безграмотно, отдельное вычисление x^{11}, притом что шаг назад был сосчитано x^{10} — это ужасно, и т.д.).

Задача 3.7 Выдача в формате программы на plain "C"

Написать программу, которая находит явный вид двух интерполирующих полиномов для функции $f(x) = 1/\cosh(x - 7/100)$. Воспользуйтесь формулой Лагранжа

$$P(x) = \sum_{i=1}^{N} f(x_i) \prod_{k=1,k\neq i}^{N} \frac{(x - x_k)}{(x_i - x_k)}$$

Для первого полинома $N = 21$, $x_i = -1.6 + 0.16 * (i - 1)$ при $i = 1 \dots 21$. Для второго — $N = 21$, $x_i = -1.6 * \cos(\pi * (i - 1)/20)$ при $i = 1 \dots 21$.

После этого она должна создать файл, который является программой на языке "plain C", работающей под `gcc`. Ручная правка полученного файла не допускается. Программа должна определять 3 функции:

`double f(double x)` — исходная функция;

`double p1(double x)` — первый интерполирующий полином;

`double p2(double x)` — второй интерполирующий полином;

Она должна сравнивать результаты интерполяции и находить максимальное отклонение от функции для первого и второго полиномов на наборе точек $t_i = -1.6 + 0.016 * (i - 1)$ при $i = 1 \dots 201$. Выходной файл, который создает программа на "C", должен содержать

$$\Delta^{(1)}_{max} \qquad \Delta^{(2)}_{max}$$

и таблицу

$$t_i \qquad f(t_i) \qquad \Delta^{(1)}_i \qquad \Delta^{(2)}_i$$

где $\Delta^{(1)}_i = f(t_i) - P_1(t_i)$, $\Delta^{(2)}_i = f(t_i) - P_2(t_i)$.

Сообразите, отчего такая разница между $\Delta^{(1)}$ и $\Delta^{(2)}$.

Замечание про вывод в файл.

Функции вывода в файл в разных версиях MAXIM'ы работают по-разному. Например, в текущей версии функция writefile почему-то делает append, а не overwrite существующего файла. Регулярный и не зависящий от версии способ выводить в файл (кстати, вывод будет идти без излишней в данном случае служебной информации) — это функция `with_stdout` в сочетании с флагом `file_output_append`:

```
file_output_append:false;
with_stdout("myfile.out", print("line 1"));
file_output_append:true;
with_stdout("myfile.out", print("line 2"));
with_stdout("myfile.out", print("line 3"));
```

(здесь первые 2 строки гарантированно затирают предыдущее содержимое файла и выводят в файл "line 1", а следующие строки позволяют дописать "line2", "line3", и т.д.)

Печать двойной кавычки и обратного слеша производится обычным образом:

```
print("Double quote: \"    ");
print("Backslash: \\    ");
```

Не мучайте компилятор `gcc` излишне длинными выражениями ("x", умноженный на себя 20 раз — это ужасно, функция `pow(x,20)` — это безграмотно, и сумма ряда, записанная в явном виде, а не в виде цикла — это ужасно, присвоения значений коэффициентов ряда при каждом вызове функции — это безграмотно, отдельное вычисление x^{11}, притом что шаг назад был сосчитано x^{10} — это ужасно, и т.д.).

Задача 3.8 Выдача в формате программы на plain "C"

Написать программу, которая находит явный вид двух интерполирующих полиномов для функции $f(x) = 1/\cosh(x - 8/100)$. Воспользуйтесь формулой Лагранжа

$$P(x) = \sum_{i=1}^{N} f(x_i) \prod_{k=1, k\neq i}^{N} \frac{(x - x_k)}{(x_i - x_k)}$$

Для первого полинома $N = 21$, $x_i = -1.6 + 0.16 * (i - 1)$ при $i = 1 \dots 21$. Для второго — $N = 21$, $x_i = -1.6 * \cos(\pi * (i - 1)/20)$ при $i = 1 \dots 21$.

После этого она должна создать файл, который является программой на языке "plain C", работающей под `gcc`. Ручная правка полученного файла не допускается. Программа должна определять 3 функции:

`double f(double x)` — исходная функция;

`double p1(double x)` — первый интерполирующий полином;

`double p2(double x)` — второй интерполирующий полином;

Она должна сравнивать результаты интерполяции и находить максимальное отклонение от функции для первого и второго полиномов на наборе точек $t_i = -1.6 + 0.016 * (i - 1)$ при $i = 1 \dots 201$. Выходной файл, который создает программа на "C", должен содержать

$$\Delta^{(1)}_{max} \qquad \Delta^{(2)}_{max}$$

и таблицу

$$t_i \qquad f(t_i) \qquad \Delta^{(1)}_i \qquad \Delta^{(2)}_i$$

где $\Delta^{(1)}_i = f(t_i) - P_1(t_i)$, $\Delta^{(2)}_i = f(t_i) - P_2(t_i)$.

Сообразите, отчего такая разница между $\Delta^{(1)}$ и $\Delta^{(2)}$.

Замечание про вывод в файл.

Функции вывода в файл в разных версиях MAXIM'ы работают по-разному. Например, в текущей версии функция writefile почему-то делает append, а не overwrite существующего файла. Регулярный и не зависящий от версии способ выводить в файл (кстати, вывод будет идти без излишней в данном случае служебной информации) — это функция `with_stdout` в сочетании с флагом `file_output_append`:

```
file_output_append:false;
with_stdout("myfile.out", print("line 1"));
file_output_append:true;
with_stdout("myfile.out", print("line 2"));
with_stdout("myfile.out", print("line 3"));
```

(здесь первые 2 строки гарантированно затирают предыдущее содержимое файла и выводят в файл "line 1", а следующие строки позволяют дописать "line2", "line3", и т.д.)

Печать двойной кавычки и обратного слеша производится обычным образом:

```
print("Double quote: \"    ");
print("Backslash: \\    ");
```

Не мучайте компилятор `gcc` излишне длинными выражениями ("x", умноженный на себя 20 раз — это ужасно, функция `pow(x,20)` — это безграмотно, и сумма ряда, записанная в явном виде, а не в виде цикла — это ужасно, присвоения значений коэффициентов ряда при каждом вызове функции — это безграмотно, отдельное вычисление x^{11}, притом что шаг назад был сосчитано x^{10} — это ужасно, и т.д.).

Задача 3.9 Выдача в формате программы на plain "C"

Написать программу, которая находит явный вид двух интерполирующих полиномов для функции $f(x) = 1/\cosh(x - 9/100)$. Воспользуйтесь формулой Лагранжа

$$P(x) = \sum_{i=1}^{N} f(x_i) \prod_{k=1, k\neq i}^{N} \frac{(x - x_k)}{(x_i - x_k)}$$

Для первого полинома $N = 21$, $x_i = -1.6 + 0.16 * (i - 1)$ при $i = 1 \dots 21$. Для второго — $N = 21$, $x_i = -1.6 * \cos(\pi * (i - 1)/20)$ при $i = 1 \dots 21$.

После этого она должна создать файл, который является программой на языке "plain C", работающей под `gcc`. Ручная правка полученного файла не допускается. Программа должна определять 3 функции:

`double f(double x)` — исходная функция;

`double p1(double x)` — первый интерполирующий полином;

`double p2(double x)` — второй интерполирующий полином;

Она должна сравнивать результаты интерполяции и находить максимальное отклонение от функции для первого и второго полиномов на наборе точек $t_i = -1.6 + 0.016 * (i - 1)$ при $i = 1 \dots 201$. Выходной файл, который создает программа на "C", должен содержать

$$\Delta^{(1)}_{max} \qquad \Delta^{(2)}_{max}$$

и таблицу

$$t_i \qquad f(t_i) \qquad \Delta^{(1)}_i \qquad \Delta^{(2)}_i$$

где $\Delta^{(1)}_i = f(t_i) - P_1(t_i)$, $\Delta^{(2)}_i = f(t_i) - P_2(t_i)$.

Сообразите, отчего такая разница между $\Delta^{(1)}$ и $\Delta^{(2)}$.

Замечание про вывод в файл.

Функции вывода в файл в разных версиях MAXIM'ы работают по-разному. Например, в текущей версии функция writefile почему-то делает append, а не overwrite существующего файла. Регулярный и не зависящий от версии способ выводить в файл (кстати, вывод будет идти без излишней в данном случае служебной информации) — это функция `with_stdout` в сочетании с флагом `file_output_append`:

```
file_output_append:false;
with_stdout("myfile.out", print("line 1"));
file_output_append:true;
with_stdout("myfile.out", print("line 2"));
with_stdout("myfile.out", print("line 3"));
```

(здесь первые 2 строки гарантированно затирают предыдущее содержимое файла и выводят в файл "line 1", а следующие строки позволяют дописать "line2", "line3", и т.д.)

Печать двойной кавычки и обратного слеша производится обычным образом:

```
print("Double quote: \"    ");
print("Backslash: \\    ");
```

Не мучайте компилятор `gcc` излишне длинными выражениями ("x", умноженный на себя 20 раз — это ужасно, функция `pow(x,20)` — это безграмотно, и сумма ряда, записанная в явном виде, а не в виде цикла — это ужасно, присвоения значений коэффициентов ряда при каждом вызове функции — это безграмотно, отдельное вычисление x^{11}, притом что шаг назад был сосчитано x^{10} — это ужасно, и т.д.).

Задача 3.10 Выдача в формате программы на plain "C"

Написать программу, которая находит явный вид двух интерполирующих полиномов для функции $f(x) = 1/\cosh(x - 10/100)$. Воспользуйтесь формулой Лагранжа

$$P(x) = \sum_{i=1}^{N} f(x_i) \prod_{k=1,k\neq i}^{N} \frac{(x - x_k)}{(x_i - x_k)}$$

Для первого полинома $N = 21$, $x_i = -1.6 + 0.16 * (i - 1)$ при $i = 1 \dots 21$. Для второго — $N = 21$, $x_i = -1.6 * \cos(\pi * (i - 1)/20)$ при $i = 1 \dots 21$.

После этого она должна создать файл, который является программой на языке "plain C", работающей под `gcc`. Ручная правка полученного файла не допускается. Программа должна определять 3 функции:

`double f(double x)` — исходная функция;

`double p1(double x)` — первый интерполирующий полином;

`double p2(double x)` — второй интерполирующий полином;

Она должна сравнивать результаты интерполяции и находить максимальное отклонение от функции для первого и второго полиномов на наборе точек $t_i = -1.6 + 0.016 * (i - 1)$ при $i = 1 \dots 201$. Выходной файл, который создает программа на "C", должен содержать

$$\Delta^{(1)}_{max} \qquad \Delta^{(2)}_{max}$$

и таблицу

$$t_i \qquad f(t_i) \qquad \Delta^{(1)}_i \qquad \Delta^{(2)}_i$$

где $\Delta^{(1)}_i = f(t_i) - P_1(t_i)$, $\Delta^{(2)}_i = f(t_i) - P_2(t_i)$.

Сообразите, отчего такая разница между $\Delta^{(1)}$ и $\Delta^{(2)}$.

Замечание про вывод в файл.

Функции вывода в файл в разных версиях MAXIM'ы работают по-разному. Например, в текущей версии функция writefile почему-то делает append, а не overwrite существующего файла. Регулярный и не зависящий от версии способ выводить в файл (кстати, вывод будет идти без излишней в данном случае служебной информации) — это функция `with_stdout` в сочетании с флагом `file_output_append`:

```
file_output_append:false;
with_stdout("myfile.out", print("line 1"));
file_output_append:true;
with_stdout("myfile.out", print("line 2"));
with_stdout("myfile.out", print("line 3"));
```

(здесь первые 2 строки гарантированно затирают предыдущее содержимое файла и выводят в файл "line 1", а следующие строки позволяют дописать "line2", "line3", и т.д.)

Печать двойной кавычки и обратного слеша производится обычным образом:

```
print("Double quote: \"    ");
print("Backslash: \\    ");
```

Не мучайте компилятор `gcc` излишне длинными выражениями ("x", умноженный на себя 20 раз — это ужасно, функция `pow(x,20)` — это безграмотно, и сумма ряда, записанная в явном виде, а не в виде цикла — это ужасно, присвоения значений коэффициентов ряда при каждом вызове функции — это безграмотно, отдельное вычисление x^{11}, притом что шаг назад был сосчитано x^{10} — это ужасно, и т.д.).