# HarrixMathLibraryForTestFunctions v.1.0

А. Б. Сергиенко

24 февраля 2014 г.

**Аннотация**

Библиотека HarrixMathLibraryForTestFunctions — это множество функций для работы с тестовыми функциями.

## Содержание

Об установке библиотеки можно прочитать тут: тут http://blog.harrix.org/?p=1279.

# 1 Введение

Библиотека HarrixMathLibraryForTestFunctions — это множество функций для работы с тестовыми функциями.

Последнюю версию документа можно найти по адресу:

https://github.com/Harrix/HarrixMathLibraryForTestFunctions

Об установке библиотеки можно прочитать тут:

http://blog.harrix.org/?p=1279

С автором можно связаться по адресу sergienkoanton@mail.ru или http://vk.com/harrix.

Сайт автора, где публикуются последние новости: http://blog.harrix.org/, а проекты располагаются по адресу http://harrix.org/.

# 2 Список функций

### Для тестовых функций

1. **MHL_ClassOfTestFunction** — Функция выдает принадлежность тестовой функции к классу функций: бинарной, вещественной или иной оптимизации.
2. **MHL_CountOfFitnessOfTestFunction_Binary** — Функция определяет количество вычислений целевой функции для тестовых задач для единообразного сравнения алгоритмов. Включает в себя все тестовые функции вещественной оптимизации.
3. **MHL_CountOfFitnessOfTestFunction_Real** — Функция определяет количество вычислений целевой функции для тестовых задач для единообразного сравнения алгоритмов. Включает в себя все тестовые функции вещественной оптимизации.
4. **MHL_DefineTestFunction** — Служебная функция определяет тестовую функцию для других функций по работе с тестовыми функциями.
5. **MHL_DimensionTestFunction_Binary** — Функция определяет размерность тестовой задачи для тестовой функции бинарной оптимизации по номеру подзадачи (число подзадач по функции MHL_GetCountOfSubProblems_Binary).
6. **MHL_DimensionTestFunction_Real** — Функция определяет размерность тестовой задачи для тестовой функции вещественной оптимизации по номеру подзадачи (число подзадач по функции MHL_GetCountOfSubProblems_Binary).
7. **MHL_ErrorExOfTestFunction_Binary** — Функция определяет значение ошибки по входным параметрам найденного решения в задаче оптимизации для тестовой функции. Включает в себя все тестовые функции бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.
8. **MHL_ErrorExOfTestFunction_Real** — Функция определяет значение ошибки по входным параметрам найденного решения в задаче оптимизации для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.
9. **MHL_ErrorEyOfTestFunction_Binary** — Функция определяет значение ошибки по значениям целевой функции найденного решения в задаче оптимизации для тестовой функции. Включает в себя все тестовые функции бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.
10. **MHL_ErrorEyOfTestFunction_Real** — Функция определяет значение ошибки по значениям целевой функции найденного решения в задаче оптимизации для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.
11. **MHL_ErrorROfTestFunction_Binary** — Функция определяет значение надежности найденного решения в задаче оптимизации для тестовой функции. Включает в себя все тестовые функции бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

12. **MHL_ErrorROfTestFunction_Real** — Функция определяет значение надежности найденного решения в задаче оптимизации для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

13. **MHL_FitnessOfOptimumOfTestFunction_Binary** — Функция определяет значение целевой функции в оптимуме для тестовой функции бинарной оптимизации. Включает в себя все тестовые функции бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

14. **MHL_FitnessOfOptimumOfTestFunction_Real** — Функция определяет значение целевой функции в оптимуме для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

15. **MHL_GetCountOfFitness** — Функция выдает количество вызовов целевой функции.

16. **MHL_GetCountOfSubProblems_Binary** — Функция определяет число подзадач (включая основную задачу) для тестовой функции бинарной оптимизации. Включает в себя все тестовые функции бинарной оптимизации.

17. **MHL_GetCountOfSubProblems_Real** — Функция определяет число подзадач (включая основную задачу) для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации.

18. **MHL_LeftBorderOfTestFunction_Real** — Функция определяет левые и правые границы допустимой области для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

19. **MHL_MaximumOrMinimumOfTestFunction_Binary** — Функция сообщает - ищется максимум или минимум в задаче оптимизации для тестовой функции бинарной оптимизации.

20. **MHL_MaximumOrMinimumOfTestFunction_Real** — Функция сообщает - ищется максимум или минимум в задаче оптимизации для тестовой функции вещественной оптимизации.

21. **MHL_NumberOfPartsOfTestFunction_Real** — Функция определяет на сколько частей нужно делить каждую координату в задаче оптимизации для тестовой функции вещественной оптимизации для алгоритма дискретной оптимизации и какая при этом требуется точность для подсчета надежности. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

22. **MHL_OptimumOfTestFunction_Binary** — Функция определяет значение оптимума для тестовой функции. Включает в себя все тестовые функции бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

23. **MHL_OptimumOfTestFunction_Real** — Функция определяет значение оптимума для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

24. **MHL_PrecisionOfCalculationsOfTestFunction_Real** — Функция определяет точность для подсчета надежности в задаче оптимизации для тестовой функции вещественной оптимизации для алгоритма дискретной оптимизации.

25. **MHL_SetToZeroCountOfFitness** — Функция обнуляет количество вызовов целевой функции. Обязательно вызвать один раз перед вызовом алгоритмов оптимизации при исследовании эффективности алгоритмов оптимизации, где требуется контроль числа вызовов целевой функции.

26. **MHL_TestFunction_Binary** — Общая тестовая функция для задач бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

27. **MHL_TestFunction_Real** — Общая тестовая функция для задач вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

# 3 Функции

## 3.1 Для тестовых функций

### 3.1.1 MHL_ClassOfTestFunction

Функция выдает принадлежность тестовой функции к классу функций: бинарной, вещественной или иной оптимизации.

Код 1. Синтаксис

```
int MHL_ClassOfTestFunction(TypeOfTestFunction Type);
```

**Входные параметры:**

Type — тип тестовой функции. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла.

**Возвращаемое значение:**

Класс тестовой функции:

1. 1 - бинарной оптимизации;
2. 2 - вещественной оптимизации.

Код 2. Пример использования

```
TypeOfTestFunction Type=TestFunction_Ackley;

//Вызов функции
int ClassOfTestFunction=MHL_ClassOfTestFunction(Type);

//используем результат
if (ClassOfTestFunction==1)
    MHL_ShowText("Это задача бинарной оптимизации");
if (ClassOfTestFunction==2)
    MHL_ShowText("Это задача вещественной оптимизации");
//Это задача вещественной оптимизации.
```

### 3.1.2 MHL_CountOfFitnessOfTestFunction_Binary

Функция определяет количество вычислений целевой функции для тестовых задач для единообразного сравнения алгоритмов. Включает в себя все тестовые функции вещественной оптимизации.

Код 3. Синтаксис

```
int MHL_CountOfFitnessOfTestFunction_Binary(int Dimension);
int MHL_CountOfFitnessOfTestFunction_Binary(int Dimension, TypeOfTestFunction Type);
```

**Входные параметры:**

Dimension — размерность тестовой задачи. Может принимать значения: 20; 30; 40; 50; 60; 70; 80; 90; 100; 200.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем.

Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Количество вычислений целевой функции для тестовых задач.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 4. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

int Dimension = 30;

//Вызов функции
int  N=MHL_CountOfFitnessOfTestFunction_Binary(Dimension);

//Использование результата
MHL_ShowNumber(N,"Количество вычислений целевой функции для TestFunction_SumVector пр
    и размерности 30","N");
//Количество вычислений целевой функции для TestFunction_SumVector при размерности
    30:
//N=400
```

### 3.1.3 MHL_CountOfFitnessOfTestFunction_Real

Функция определяет количество вычислений целевой функции для тестовых задач для единообразного сравнения алгоритмов. Включает в себя все тестовые функции вещественной оптимизации.

Код 5. Синтаксис

```
int MHL_CountOfFitnessOfTestFunction_Real(int Dimension);
int MHL_CountOfFitnessOfTestFunction_Real(int Dimension, TypeOfTestFunction Type);
```

**Входные параметры:**

Dimension — размерность тестовой задачи. Может принимать значения: 2; 3; 4; 5; 10; 20; 30.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем.

Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Количество вычислений целевой функции для тестовых задач.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 6. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

int Dimension = 3;

//Вызов функции
int  N=MHL_CountOfFitnessOfTestFunction_Real(Dimension);

//Использование результата
MHL_ShowNumber(N,"Количество вычислений целевой функции для TestFunction_Ackley при р
    азмерности 3","N");
//Количество вычислений целевой функции для TestFunction_Ackley при размерности 3:
//N=729
```

### 3.1.4 MHL_DefineTestFunction

Служебная функция определяет тестовую функцию для других функций по работе с тестовыми функциями.

Код 7. Синтаксис

```
void MHL_DefineTestFunction(TypeOfTestFunction Type);
```

**Входные параметры:**

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Отсутствует.

Код 8. Пример использования

```
//Вызов функции
MHL_DefineTestFunction(TestFunction_SumVector);

//Использование результата
int N=5;
int *x=new int[N];
TMHL_RandomBinaryVector(x,N);
double f=MHL_TestFunction_Binary(x,N);

MHL_ShowVectorT(x,N,"Решение","x");
//Решение:
//x =
//1   1   1   1   0

MHL_ShowNumber(f,"Значение целевой функции","f");
```

```
//Значение целевой функции:
//f=4
```

### 3.1.5 MHL_DimensionTestFunction_Binary

Функция определяет размерность тестовой задачи для тестовой функции бинарной оптимизации по номеру подзадачи (число подзадач по функции MHL_GetCountOfSubProblems_Binary).

Код 9. Синтаксис

```
int MHL_DimensionTestFunction_Binary(int i);
int MHL_DimensionTestFunction_Binary(int i, TypeOfTestFunction Type);
```

**Входные параметры:**

**i** - номер подзадачи (начиная с нуля).

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем.

Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Размерность тестовой задачи для тестовой функции бинарной оптимизации.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 10. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

//Вызов функции
double N=MHL_DimensionTestFunction_Binary(0);

//Использование результата
MHL_ShowNumber(N,"Размерность тестовой задачи для TestFunction_SumVector при i=0","N"
    );
//Размерность тестовой задачи для TestFunction_SumVector при i=0:
//N=20
```

### 3.1.6 MHL_DimensionTestFunction_Real

Функция определяет размерность тестовой задачи для тестовой функции вещественной оптимизации по номеру подзадачи (число подзадач по функции MHL_GetCountOfSubProblems_Binary).

Код 11. Синтаксис

```
int MHL_DimensionTestFunction_Real(int i);
int MHL_DimensionTestFunction_Real(int i, TypeOfTestFunction Type);
```

**Входные параметры:**

**i** - номер подзадачи (начиная с нуля).

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем.

Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Размерность тестовой задачи для тестовой функции вещественной оптимизации.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 12. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

//Вызов функции
double N=MHL_DimensionTestFunction_Real(0);

//Использование результата
MHL_ShowNumber(N,"Размерность тестовой задачи для TestFunction_Ackley при i=0","N");
//Размерность тестовой задачи для TestFunction_Ackley при i=0:
//N=2
```

### 3.1.7 MHL_ErrorExOfTestFunction_Binary

Функция определяет значение ошибки по входным параметрам найденного решения в задаче оптимизации для тестовой функции. Включает в себя все тестовые функции бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 13. Синтаксис

```
double MHL_ErrorExOfTestFunction_Binary(int *x, int VMHL_N);
double MHL_ErrorExOfTestFunction_Binary(int *x, int VMHL_N, TypeOfTestFunction Type);
```

**Входные параметры:**

x — указатель на исходный массив (найденное решение алгоритмом);

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley,

TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение ошибки по входным параметрам Ex.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Конкретную формулу, которые используются для нахождения для каждой тестовой функции, смотрите в функциях этих тестовых функций. Обратите внимание, что данная функция находит ошибку только для одного решения, тогда как по формулам нужно множество решений.

Код 14. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

int N=5;
int *x=new int[N];
TMHL_RandomBinaryVector(x,N);

//Вызов функции
double Ex=MHL_ErrorExOfTestFunction_Binary(x,N);

//Использование результата
MHL_ShowVectorT(x,N,"Решение","x");
//Решение:
//x =
//1   0   1   1   1

MHL_ShowNumber(Ex,"Значение ошибки по входным параметрам","E<sub>x</sub>");
//Значение ошибки по входным параметрам:
//Ex=1
```

### 3.1.8 MHL_ErrorExOfTestFunction_Real

Функция определяет значение ошибки по входным параметрам найденного решения в задаче оптимизации для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 15. Синтаксис

```
double MHL_ErrorExOfTestFunction_Real(double *x, int VMHL_N);
double MHL_ErrorExOfTestFunction_Real(double *x, int VMHL_N, TypeOfTestFunction Type)
   ;
```

**Входные параметры:**

x — указатель на исходный массив (найденное решение алгоритмом);

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение ошибки по входным параметрам Ex.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Конкретную формулу, которые используются для нахождения для каждой тестовой функции, смотрите в функциях этих тестовых функций. Обратите внимание, что данная функция находит ошибку только для одного решения, тогда как по формулам нужно множество решений.

Код 16. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

int N=5;
double *x=new double[N];
MHL_RandomRealVector(x,-0.5,0.05,N);

//Вызов функции
double Ex=MHL_ErrorExOfTestFunction_Real(x,N);

//Использование результата
MHL_ShowVectorT(x,N,"Решение","x");
//Решение:
//x =
//-0.43694  -0.458693   -0.0266388  0.0117142   -0.136948


MHL_ShowNumber(Ex,"Значение ошибки по входным параметрам","E<sub>x</sub>");
//Значение ошибки по входным параметрам:
//Ex=0.129756
```

### 3.1.9 MHL_ErrorEyOfTestFunction_Binary

Функция определяет значение ошибки по значениям целевой функции найденного решения в задаче оптимизации для тестовой функции. Включает в себя все тестовые функции бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 17. Синтаксис

```
double MHL_ErrorEyOfTestFunction_Binary(double FitnessOfx, int VMHL_N);
double MHL_ErrorEyOfTestFunction_Binary(double FitnessOfx, int VMHL_N,
   TypeOfTestFunction Type);
```

**Входные параметры:**

FitnessOfx — значение целевой функции найденного решения алгоритмом оптимизации;

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение ошибки по значениям целевой функции Ey.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Конкретную формулу, которые используются для нахождения для каждой тестовой функции, смотрите в функциях этих тестовых функций. Обратите внимание, что данная функция находит ошибку только для одного решения, тогда как по формулам нужно множество решений.

Все функции так высчитываются, чтобы алгоритм решал задачу поиска максимального значения целевой функции, поэтому тестовые функции на минимум умножаются на $-1$. Поэтому, фактически алгоритмы оптимизации находят максимум перевернутой функции. А значит, чтобы правильно посчитать ошибку по значениям целевой функции, нужно найденное решение умножить на $-1$.

Код 18. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

int N=5;
int *x=new int[N];
TMHL_RandomBinaryVector(x,N);
double f=MHL_TestFunction_Binary(x,N);

//Вызов функции
double Ey=MHL_ErrorEyOfTestFunction_Binary(f,N);

//Использование результата
MHL_ShowVectorT(x,N,"Решение","x");
//Решение:
//x =
//0   1   1   0   1

MHL_ShowNumber(Ey,"Значение ошибки по значениям целевой функции","E<sub>y</sub>");
//Значение ошибки по значениям целевой функции:
//Ey=2
```

### 3.1.10 MHL_ErrorEyOfTestFunction_Real

Функция определяет значение ошибки по значениям целевой функции найденного решения в задаче оптимизации для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 19. Синтаксис

```
double MHL_ErrorEyOfTestFunction_Real(double FitnessOfx, int VMHL_N);
double MHL_ErrorEyOfTestFunction_Real(double FitnessOfx, int VMHL_N,
    TypeOfTestFunction Type);
```

**Входные параметры:**

FitnessOfx — значение целевой функции найденного решения алгоритмом оптимизации;

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение ошибки по значениям целевой функции Ey.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Конкретную формулу, которые используются для нахождения для каждой тестовой функции, смотрите в функциях этих тестовых функций. Обратите внимание, что данная функция находит ошибку только для одного решения, тогда как по формулам нужно множество решений.

Все функции так высчитываются, чтобы алгоритм решал задачу поиска максимального значения целевой функции, поэтому тестовые функции на минимум умножаются на $-1$. Поэтому, фактически алгоритмы оптимизации находят максимум перевернутой функции. А значит, чтобы правильно посчитать ошибку по значениям целевой функции, нужно найденное решение умножить на $-1$.

Код 20. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

int N=5;
double *x=new double[N];
MHL_RandomRealVector(x,-0.5,0.05,N);
double f=MHL_TestFunction_Real(x,N);

//Вызов функции
double Ey=MHL_ErrorEyOfTestFunction_Real(f,N);

//Использование результата
MHL_ShowVectorT(x,N,"Решение","x");
//Решение:
//x =
//-0.0963959   -0.183693   -0.0485428  -0.185757   0.0321075

MHL_ShowNumber(Ey,"Значение ошибки по значениям целевой функции","E<sub>y</sub>");
//Значение ошибки по значениям целевой функции:
//Ey=1.18549ы
```

### 3.1.11 MHL_ErrorROfTestFunction_Binary

Функция определяет значение надежности найденного решения в задаче оптимизации для тестовой функции. Включает в себя все тестовые функции бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 21. Синтаксис

```
double MHL_ErrorROfTestFunction_Binary(int *x, int VMHL_N);
double MHL_ErrorROfTestFunction_Binary(int *x, int VMHL_N, TypeOfTestFunction Type);
```

**Входные параметры:**

x — указатель на исходный массив (найденное решение алгоритмом);

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение надежности R.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Конкретную формулу, которые используются для нахождения для каждой тестовой функции, смотрите в функциях этих тестовых функций. Обратите внимание, что данная функция находит ошибку только для одного решения, тогда как по формулам нужно множество решений.

Код 22. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

int N=5;
int *x=new int[N];
TMHL_RandomBinaryVector(x,N);

//Вызов функции
double R=MHL_ErrorROfTestFunction_Binary(x,N);

//Использование результата
MHL_ShowVectorT(x,N,"Решение","x");
//Решение:
//x =
//1   1   1   1   1

MHL_ShowNumber(R,"Значение надежности","R");
//Значение надежности:
//R=1
```

### 3.1.12 MHL_ErrorROfTestFunction_Real

Функция определяет значение надежности найденного решения в задаче оптимизации для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 23. Синтаксис

```
double MHL_ErrorROfTestFunction_Real(double *x, int VMHL_N);
double MHL_ErrorROfTestFunction_Real(double *x, int VMHL_N, TypeOfTestFunction Type);
```

**Входные параметры:**

x — указатель на исходный массив (найденное решение алгоритмом);

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение надежности R.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Конкретную формулу, которые используются для нахождения для каждой тестовой функции, смотрите в функциях этих тестовых функций. Обратите внимание, что данная функция находит ошибку только для одного решения, тогда как по формулам нужно множество решений.

Код 24. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

int N=5;
double *x=new double[N];
MHL_RandomRealVector(x,0.01,0.02,N);

//Вызов функции
double R=MHL_ErrorROfTestFunction_Real(x,N);

//Использование результата
MHL_ShowVectorT(x,N,"Решение","x");
//Решение:
//x =
//0.0118939 0.0177618   0.0115656   0.0181937   0.0124084

MHL_ShowNumber(R,"Значение надежности","R");
//Значение надежности:
//R=1
```

### 3.1.13 MHL_FitnessOfOptimumOfTestFunction_Binary

Функция определяет значение целевой функции в оптимуме для тестовой функции бинарной оптимизации. Включает в себя все тестовые функции бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 25. Синтаксис

```
double MHL_FitnessOfOptimumOfTestFunction_Binary(int VMHL_N);
double MHL_FitnessOfOptimumOfTestFunction_Binary(int VMHL_N, TypeOfTestFunction Type)
    ;
```

**Входные параметры:**

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение тестовой функции в оптимальной точке.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 26. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

int N=5;

//Вызов функции
double f=MHL_FitnessOfOptimumOfTestFunction_Binary(N);

//Использование результата
MHL_ShowNumber(f,"Значение целевой функции оптимального решения","f");
//Значение целевой функции оптимального решения:
//f=5
```

### 3.1.14 MHL_FitnessOfOptimumOfTestFunction_Real

Функция определяет значение целевой функции в оптимуме для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 27. Синтаксис

```
double MHL_FitnessOfOptimumOfTestFunction_Real(double VMHL_N);
double MHL_FitnessOfOptimumOfTestFunction_Real(double VMHL_N, TypeOfTestFunction Type
    );
```

**Входные параметры:**

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение тестовой функции в оптимальной точке.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 28. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

int N=5;

//Вызов функции
double f=MHL_FitnessOfOptimumOfTestFunction_Binary(N);

//Использование результата
MHL_ShowNumber(f,"Значение целевой функции оптимального решения функции
    TestFunction_Ackley","f");
//Значение целевой функции оптимального решения функции TestFunction_Ackley:
//f=0
```

### 3.1.15 MHL_GetCountOfFitness

Функция выдает количество вызовов целевой функции.

Код 29. Синтаксис

```
int MHL_GetCountOfFitness();
```

**Входные параметры:**

Отсутствуют.

**Возвращаемое значение:**

Количество вызовов целевой функции.

Данную функцию надо использовать в связке с функцией MHL_GetCountOfFitness(). Для чего использовать эти функции? Дело в том, что для сравнения алгоритмов оптимизации очень критично оценивать вызов целевой функции. И часто многие программисты пишут или не очень акккуратно, или логика алгоритма такая, что заявленное число вычислений функций не совпадает с действительным. Поэтому в общие тестовые функции (например, MHL_TestFunction_Binary) вшит подсчет числа вызовов целевой функции.

MHL_SetToZeroCountOfFitness — эту функцию вызываем перед вызовом какого-то алгоритма оптимизации, а MHL_GetCountOfFitness — после его работы.

Код 30. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

MHL_SetToZeroCountOfFitness();

int N=5;
double f=0;
int *x=new int[N];

for (int i=0;i<10;i++)
{
    TMHL_RandomBinaryVector(x,N);
    f+=MHL_TestFunction_Binary(x,N);
}

f/=double(10.);

//Вызов функции
int M=MHL_GetCountOfFitness();

//Использование результата
MHL_ShowNumber(M,"Количество вызовов целевой функции","M");
//Количество вызовов целевой функции:
//M=10

MHL_ShowNumber(f,"Среднее значение целевой функции","f");
//Среднее значение целевой функции:
//f=2.6
```

### 3.1.16 MHL_GetCountOfSubProblems_Binary

Функция определяет число подзадач (включая основную задачу) для тестовой функции бинарной оптимизации. Включает в себя все тестовые функции бинарной оптимизации.

Код 31. Синтаксис

```
int MHL_GetCountOfSubProblems_Binary();
int MHL_GetCountOfSubProblems_Binary(TypeOfTestFunction Type);
```

**Входные параметры:**

Отсутствуют.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем.

Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Число подзадач (включая основную задачу) для тестовой функции.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 32. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

//Вызов функции
double N=MHL_GetCountOfSubProblems_Binary();

//Использование результата
MHL_ShowNumber(N,"Число подзадач (включая основную задачу) для TestFunction_SumVector
    ","N");
//Число подзадач (включая основную задачу) для TestFunction_SumVector:
//N=10
```

### 3.1.17 MHL_GetCountOfSubProblems_Real

Функция определяет число подзадач (включая основную задачу) для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации.

Код 33. Синтаксис

```
int MHL_GetCountOfSubProblems_Real();
int MHL_GetCountOfSubProblems_Real(TypeOfTestFunction Type);
```

**Входные параметры:**

Отсутствуют.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем.

Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Число подзадач (включая основную задачу) для тестовой функции.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 34. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

//Вызов функции
double N=MHL_GetCountOfSubProblems_Real();

//Использование результата
MHL_ShowNumber(N,"Число подзадач (включая основную задачу) для TestFunction_Ackley","
    N");
//Число подзадач (включая основную задачу) для TestFunction_Ackley:
```

```
//N=7
```

### 3.1.18 MHL_LeftBorderOfTestFunction_Real

Функция определяет левые и правые границы допустимой области для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 35. Синтаксис

```
void MHL_LeftAndRightBorderOfTestFunction_Real(double *Left, double *Right,int VMHL_N
    );
void MHL_LeftAndRightBorderOfTestFunction_Real(double *Left, double *Right, int
    VMHL_N, TypeOfTestFunction Type);
```

**Входные параметры:**

Left — указатель на массив, куда будет записываться результат левых границ допустимой области;

Right — указатель на массив, куда будет записываться результат левых границ допустимой области;

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Отсутствует.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 36. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

int N=5;
double *Left=new double[N];
double *Right=new double[N];

//Вызов функции
MHL_LeftAndRightBorderOfTestFunction_Real(Left,Right,N);

//Использование результата
MHL_ShowVectorT(Left,N,"Левые границы допустимой области функции TestFunction_Ackley"
    ,"Left");
//Левые границы допустимой области функции TestFunction_Ackley:
//Left =
//-5  -5 -5 -5 -5
```

```
MHL_ShowVectorT(Right,N,"Правые границы допустимой области функции
    TestFunction_Ackley","Right");
//Правые границы допустимой области функции TestFunction_Ackley:
//Right =
//5   5  5  5  5
```

### 3.1.19 MHL_MaximumOrMinimumOfTestFunction_Binary

Функция сообщает - ищется максимум или минимум в задаче оптимизации для тестовой функции бинарной оптимизации.

Код 37. Синтаксис

```
double MHL_MaximumOrMinimumOfTestFunction_Binary();
double MHL_MaximumOrMinimumOfTestFunction_Binary(TypeOfTestFunction Type);
```

**Входные параметры:**

Отсутствуют.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

1 — задача на нахождение максимума;

-1 — задача на нахождение минимума.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 38. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

//Вызов функции
double MorM=MHL_MaximumOrMinimumOfTestFunction_Binary();

//Использование результата
MHL_ShowNumber(MorM,"Максимум или минимум функции находим у TestFunction_SumVector","
    MorM");
//Максимум или минимум функции находим у TestFunction_SumVector:
//MorM=1
```

### 3.1.20 MHL_MaximumOrMinimumOfTestFunction_Real

Функция сообщает - ищется максимум или минимум в задаче оптимизации для тестовой функции вещественной оптимизации.

Код 39. Синтаксис

```
double MHL_MaximumOrMinimumOfTestFunction_Real();
double MHL_MaximumOrMinimumOfTestFunction_Real(TypeOfTestFunction Type);
```

**Входные параметры:**

Отсутствуют.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

1 — задача на нахождение максимума;

-1 — задача на нахождение минимума.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 40. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

//Вызов функции
double MorM=MHL_MaximumOrMinimumOfTestFunction_Real();

//Использование результата
MHL_ShowNumber(MorM,"Максимум или минимум функции находим у TestFunction_Ackley","
    MorM");
//Максимум или минимум функции находим у TestFunction_Ackley:
//MorM=-1
```

### 3.1.21 MHL_NumberOfPartsOfTestFunction_Real

Функция определяет на сколько частей нужно делить каждую координату в задаче оптимизации для тестовой функции вещественной оптимизации для алгоритма дискретной оптимизации и какая при этом требуется точность для подсчета надежности. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 41. Синтаксис

```
double MHL_NumberOfPartsOfTestFunction_Real(int *NumberOfParts, int VMHL_N);
double MHL_NumberOfPartsOfTestFunction_Real(int *NumberOfParts, int VMHL_N,
    TypeOfTestFunction Type);
```

**Входные параметры:**

NumberOfParts — указатель на массив, куда будет записываться результат;

VMHL_N — размер массива NumberOfParts.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Точность вычислений.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 42. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

int N=5;
int *NumberOfParts=new int[N];

//Вызов функции
double e=MHL_NumberOfPartsOfTestFunction_Real(NumberOfParts,N);

//Использование результата
MHL_ShowVectorT(NumberOfParts,N,"На сколько частей нужно делить каждую координату фун
    кции TestFunction_Ackley","NumberOfParts");
//На сколько частей нужно делить каждую координату функции TestFunction_Ackley:
//NumberOfParts =
//4095   4095  4095  4095  4095

MHL_ShowNumber(e,"Точность вычислений.","e");
//Точность вычислений.:
//e=0.025
```

### 3.1.22 MHL_OptimumOfTestFunction_Binary

Функция определяет значение оптимума для тестовой функции. Включает в себя все тестовые функции бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 43. Синтаксис

```
double MHL_OptimumOfTestFunction_Binary(int *Optimum, int VMHL_N);
double MHL_OptimumOfTestFunction_Binary(int *Optimum, int VMHL_N, TypeOfTestFunction
    Type);
```

**Входные параметры:**

Optimum — указатель на исходный массив, куда будет записываться результат, то есть оптимум тестовой функции (максимум или минимум — это зависит от типа тестовой функции, что расписывается в самих функциях тестовых функций);

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение тестовой функции в оптимальной точке.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 44. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

int N=5;
int *x=new int[N];

//Вызов функции
double f=MHL_OptimumOfTestFunction_Binary(x,N);

//Использование результата
MHL_ShowVectorT(x,N,"Оптимальное решение тестовой функции TestFunction_SumVector","x"
    );
//Оптимальное решение тестовой функции TestFunction_SumVector:
//x =
//1   1   1   1   1

MHL_ShowNumber(f,"Значение целевой функции оптимального решения","f");
//Значение целевой функции оптимального решения:
//f=5
```

### 3.1.23 MHL_OptimumOfTestFunction_Real

Функция определяет значение оптимума для тестовой функции вещественной оптимизации. Включает в себя все тестовые функции вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 45. Синтаксис

```
double MHL_OptimumOfTestFunction_Real(double *Optimum, int VMHL_N);
double MHL_OptimumOfTestFunction_Real(double *Optimum, int VMHL_N, TypeOfTestFunction
    Type);
```

**Входные параметры:**

Optimum — указатель на исходный массив, куда будет записываться результат, то есть оптимум тестовой функции (максимум или минимум — это зависит от типа тестовой функции, что расписывается в самих функциях тестовых функций);

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение тестовой функции в оптимальной точке.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 46. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

int N=5;
double *x=new double[N];

//Вызов функции
double f=MHL_OptimumOfTestFunction_Real(x,N);

//Использование результата
MHL_ShowVectorT(x,N,"Оптимальное решение тестовой функции TestFunction_Ackley","x");
//Оптимальное решение тестовой функции TestFunction_Ackley:
//x =
//0   0   0   0   0

MHL_ShowNumber(f,"Значение целевой функции оптимального решения","f");
//Значение целевой функции оптимального решения:
//f=0
```

### 3.1.24 MHL_PrecisionOfCalculationsOfTestFunction_Real

Функция определяет точность для подсчета надежности в задаче оптимизации для тестовой функции вещественной оптимизации для алгоритма дискретной оптимизации.

Код 47. Синтаксис

```
double MHL_PrecisionOfCalculationsOfTestFunction_Real();
double MHL_PrecisionOfCalculationsOfTestFunction_Real(TypeOfTestFunction Type);
```

**Входные параметры:**

Отсутствуют.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Точность вычислений.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Код 48. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

//Вызов функции
double e=MHL_PrecisionOfCalculationsOfTestFunction_Real();

//Использование результата
MHL_ShowNumber(e,"Точность вычислений","e");
//Точность вычислений:
//e=0.025
```

### 3.1.25 MHL_SetToZeroCountOfFitness

Функция обнуляет количество вызовов целевой функции. Обязательно вызвать один раз перед вызовом алгоритмов оптимизации при исследовании эффективности алгоритмов оптимизации, где требуется контроль числа вызовов целевой функции.

Код 49. Синтаксис

```
void MHL_SetToZeroCountOfFitness();
```

**Входные параметры:**

Отсутствуют.

**Возвращаемое значение:**

Отсутствует.

Данную функцию надо использовать в связке с функцией MHL_GetCountOfFitness(). Для чего использовать эти функции? Дело в том, что для сравнения алгоритмов оптимизации очень критично оценивать вызов целевой функции. И часто многие программисты пишут или не очень акккуратно, или логика алгоритма такая, что заявленное число вычислений функций не совпадает с действительным. Поэтому в общие тестовые функции (например, MHL_TestFunction_Binary) вшит подсчет числа вызовов целевой функции.

MHL_SetToZeroCountOfFitness — эту функцию вызываем перед вызовом какого-то алгоритма оптимизации, а MHL_GetCountOfFitness — после его работы.

Код 50. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

//Вызов функции
MHL_SetToZeroCountOfFitness();

//Использование результата
int N=5;
double f=0;
int *x=new int[N];

for (int i=0;i<10;i++)
```

```
{
    TMHL_RandomBinaryVector(x,N);
    f+=MHL_TestFunction_Binary(x,N);
}

f/=double(10.);

int M=MHL_GetCountOfFitness();
MHL_ShowNumber(M,"Количество вызовов целевой функции","M");
//Количество вызовов целевой функции:
//M=10

MHL_ShowNumber(f,"Среднее значение целевой функции","f");
//Среднее значение целевой функции:
//f=2.6
```

### 3.1.26 MHL_TestFunction_Binary

Общая тестовая функция для задач бинарной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 51. Синтаксис

```
double MHL_TestFunction_Binary(int *x, int VMHL_N);
double MHL_TestFunction_Binary(int *x, int VMHL_N, TypeOfTestFunction Type);
```

**Входные параметры:**

x — указатель на исходный массив;

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение тестовой функции в точке x.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Все функции так высчитываются, чтобы алгоритм решал задачу поиска максимального значения целевой функции, поэтому тестовые функции на минимум умножаются на $-1$.

Код 52. Пример использования

```
MHL_DefineTestFunction(TestFunction_SumVector);

int N=5;
int *x=new int[N];
TMHL_RandomBinaryVector(x,N);
```

```
//Вызов функции
double f=MHL_TestFunction_Binary(x,N);

//Использование результата
MHL_ShowVectorT(x,N,"Решение","x");
//Решение:
//x =
//1   1   1   1   0

MHL_ShowNumber(f,"Значение целевой функции","f");
//Значение целевой функции:
//f=4
```

### 3.1.27 MHL_TestFunction_Real

Общая тестовая функция для задач вещественной оптимизации. Есть функция-переопределение, где пользователь может сам указать тип тестовой функции.

Код 53. Синтаксис

```
double MHL_TestFunction_Real(double *x, int VMHL_N);
double MHL_TestFunction_Real(double *x, int VMHL_N, TypeOfTestFunction Type);
```

**Входные параметры:**

x — указатель на исходный массив;

VMHL_N — размер массива x.

В переопределяемой функции также есть параметр:

Type — обозначение тестовой функции, которую вызываем. Смотреть виды в переменных перечисляемого типа в начале HarrixMathLibrary.h файла: TestFunction_Ackley, TestFunction_ParaboloidOfRevolution, TestFunction_Rastrigin и др. Они совпадают с названиями одноименных тестовых функций, но без приставки **MHL_**.

**Возвращаемое значение:**

Значение тестовой функции в точке x.

Итак, для обычного использования (без параметра Type) нужно вызвать функцию MHL_DefineTestFunction. Иначе использовать переопределенную функцию и самому указать тип тестовой функции.

Все функции так высчитываются, чтобы алгоритм решал задачу поиска максимального значения целевой функции, поэтому тестовые функции на минимум умножаются на −1.

Код 54. Пример использования

```
MHL_DefineTestFunction(TestFunction_Ackley);

int N=5;
double *x=new double[N];
MHL_RandomRealVector(x,-1,1,N);

//Вызов функции
double f=MHL_TestFunction_Real(x,N);
```

```
//Использование результата
MHL_ShowVectorT(x,N,"Решение","x");
//Решение:
//x =
//-0.391724 0.347656 0.259155 -0.544617   0.116516

MHL_ShowNumber(f,"Значение целевой функции","f");
//Значение целевой функции:
//f=3.38932
```