
Справочник API

Выпуск 2.0.11.135584

дек. 20, 2024

1	C++ API	1
2	C# API	21
3	Python API	40
4	Параметры	62
	Содержание модулей Python	65
	Алфавитный указатель	66

```
class arhiplex_exception : public runtime_error
```

arhiplex-исключение

Исключение, генерируемое всеми классами в случае ошибки

Public Functions

```
inline int get_error_code() const
```

Код ошибки arhiplex. Как правило, всегда -1

```
class Constraint
```

ограничение

Работа с ограничениями у модели, позволяет работать с выражением и границами а также удалять его

Public Functions

```
inline Constraint(const LinearExpression &expr, constraint_sense sense, double rhs)
```

создает ограничение вида (expr (\leq , $=$, \geq) constant)

Параметры

- **expr** – [in] выражение, являющееся левой частью ограничения
- **sense** – [in] переменная, задающая знак (\leq , $=$, \geq)
- **rhs** – [in] правая часть ограничения

```
inline Constraint(const QuadExpression &expr, constraint_sense sense, double rhs)
```

создает ограничение вида (expr (\leq , $=$, \geq) constant)

Параметры

- **expr** – **[in]** выражение, являющееся левой частью ограничения
- **sense** – **[in]** переменная, задающая знак (\leq , $=$, \geq)
- **rhs** – **[in]** правая часть ограничения

inline *LinearExpression* GetLinearExpression() const

Линейное выражение, связанное с ограничением

Результат

Выражение, связанное с ограничением

inline *QuadExpression* GetQuadExpression() const

Квадратичное выражение, связанное с ограничением

Результат

Выражение, связанное с ограничением

inline void SetUpperBound(double upper_bound)

Задаёт верхнюю границу ограничения

Параметры

upper_bound – **[in]** Новая верхняя граница ограничения

inline double GetUpperBound() const

Получает верхнюю границу ограничения

Результат

Верхнюю границу ограничения

inline void SetLowerBound(double lower_bound)

Задаёт нижнюю границу ограничения

Параметры

lower_bound – **[in]** Новая нижняя граница ограничения

inline double GetLowerBound() const

Получает нижнюю границу ограничения

Результат

Нижнюю границу ограничения

inline double GetRange() const

Получает интервал ограничения: upper_bound - lower_bound

Результат

upper_bound - lower_bound

inline void SetRange(double range)

Задаёт интервал ограничения: lower_bound \leq expression \leq lower_bound + range

Параметры

range – **[in]** интервал для ограничения

`inline void Remove()`

Помечает ограничение как удаленное. Такое ограничение будет исключено из расчетов

`inline const char *GetName() const`

Получает имя ограничения

Результат

имя ограничения

`inline void SetName(const char *szName)`

Задаёт имя ограничения

Параметры

`szName` – **[in]** имя ограничения

`class LinearExpression`

Линейное выражение

Обеспечивает работу с линейными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная * коэффициент.

Public Functions

`inline LinearExpression(double constant = 0.0)`

Создает пустое выражение с возможностью задать константу

Параметры

`constant` – **[in]** константа в выражении

`inline LinearExpression(const Variable &var, double coeff = 1.0)`

Создает выражение на основе переменной и коэффициента

Параметры

- `var` – **[in]** переменная в выражении
- `coeff` – **[in]** коэффициент переменной в выражении

`inline int GetTermsCount() const`

Возвращает кол-во элементов в выражении

Результат

Кол-во элементов в выражении

`inline bool empty() const`

Возвращает истину, если выражение пустое

Результат

истина, если выражение пустое

`inline void SetName(const char *expr_name)`

Задаёт имя выражения

Параметры

`expr_name` – **[in]** имя выражения

inline std::string GetName() const

Получает имя выражения

Результат

имя выражения

inline *Variable* GetTermVariable(int i) const

Возвращает переменную по индексу элемента в выражении

Параметры

i – [in] индекс элемента в выражении

Результат

Объект переменной

inline double GetTermCoeff(int i) const

Возвращает коэффициент переменной по индексу элемента в выражении

Параметры

i – [in] индекс элемента в выражении

Результат

Значение коэффициента переменной

inline double GetConstant() const

Возвращает константу в выражении

Результат

значение константы

inline void SetTermCoeff(int i, double value)

Задаёт коэффициент переменной по индексу выражения

Параметры

- *i* – [in] Индекс элемента в выражении
- *value* – [in] Значение коэффициента при переменной в элементе

inline void SetConstant(double constant)

Задаёт константу в выражении

Параметры

constant – [in] значение константы

inline void AddConstant(double constant)

Добавляет константу к выражению

Параметры

constant – [in] значение константы

inline void AddTerm(const *Variable* &var, double coeff = 1.0)

Добавляет элемент к выражению

Параметры

- *var* – [in] переменная
- *coeff* – [in] коэффициент к переменной

```
inline void AddExpression(const LinearExpression &expr, double mult = 1.0)
```

Добавляет выражение к выражению

Параметры

- **expr** – **[in]** добавляемое выражение
- **mult** – **[in]** коэффициент к добавляемому выражению

```
inline void RemoveTerm(int idx)
```

Удаляет элемент выражения по индексу

Параметры

idx – **[in]** индекс удаляемого элемента в выражении

```
inline void RemoveVariable(const Variable &var)
```

Удаляет переменную из выражения

Параметры

var – **[in]** удаляемая переменная

```
inline LinearExpression CreateFreeCopy() const
```

Создает копию выражения, не привязанную к модели или другому ограничению

Результат

копия выражения

```
class QuadExpression
```

Линейное выражение

Обеспечивает работу с квадратичными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная * переменная * коэффициент.

Public Functions

```
inline QuadExpression()
```

Создает пустое выражение

```
inline QuadExpression(const Variable &var1, const Variable &var2, double coeff = 1.0)
```

Создает выражение на основе переменных и коэффициента

Параметры

- **var1** – **[in]** переменная №1 в выражении
- **var2** – **[in]** переменная №1 в выражении
- **coeff** – **[in]** коэффициент в выражении

```
inline int GetTermsCount() const
```

Возвращает кол-во элементов в выражении

Результат

Кол-во элементов в выражении

inline bool **empty()** const

Возвращает истину, если выражение пустое

Результат

истина, если выражение пустое

inline *Variable* **GetTermVariable1**(int i) const

Возвращает переменную №1 по индексу элемента в выражении

Параметры

i – [in] индекс элемента в выражении

Результат

Объект переменной

inline *Variable* **GetTermVariable2**(int i) const

Возвращает переменную №1 по индексу элемента в выражении

Параметры

i – [in] индекс элемента в выражении

Результат

Объект переменной

inline double **GetTermCoeff**(int i) const

Возвращает коэффициент переменных по индексу элемента в выражении

Параметры

i – [in] индекс элемента в выражении

Результат

Значение коэффициента

inline void **SetTermCoeff**(int i, double value)

Задаёт коэффициент переменной по индексу выражения

Параметры

- i – [in] Индекс элемента в выражении
- value – [in] Значение коэффициента при переменных в элементе

inline void **AddTerm**(const *Variable* &var1, const *Variable* &var2, double coeff = 1.0)

Добавляет элемент к выражению

Параметры

- var1 – [in] переменная
- var2 – [in] переменная
- coeff – [in] коэффициент

inline void **AddExpression**(const *QuadExpression* &expr, double mult = 1.0)

Добавляет выражение к выражению

Параметры

- expr – [in] квадратичное выражение
- mult – [in] коэффициент к добавляемому выражению


```
inline void AddExpression(const LinearExpression &expr, double mult = 1.0)
```

Добавляет выражение к выражению

Параметры

- **expr** – **[in]** добавляемое выражение
- **mult** – **[in]** коэффициент к добавляемому выражению

```
inline void RemoveTerm(int idx)
```

Удаляет элемент выражения по индексу

Параметры

idx – **[in]** индекс удаляемого элемента в выражении

```
inline QuadExpression CreateFreeCopy() const
```

Создает копию выражения, не привязанную к модели или другому ограничению

Результат

копия выражения

```
class Model
```

Класс для работы с моделью

Предоставляет функции чтения/записи файлов, модификации модели, управления переменными, ограничениями, а также целевой функцией

Public Functions

```
inline Model(const char *name = nullptr)
```

Создает экземпляр модели

Параметры

name – **[in]** Имя модели

```
inline void Read(const char *szFileName)
```

Читает модель из файла, тип модели определяется по расширению

Параметры

szFileName – **[in]** путь к файлу модели

```
inline void ReadMps(const char *szFileName)
```

Читает модель из файла, при этом он будет читаться как MPS файл независимо от расширения

Параметры

szFileName – **[in]** путь к файлу модели

```
inline void ReadLp(const char *szFileName)
```

Читает модель из файла, при этом он будет читаться как LP файл независимо от расширения

Параметры

szFileName – **[in]** путь к файлу модели

```
inline int GetIntParam(const char *szParam)
```

Получает целочисленный параметр

Параметры

`szParam` – **[in]** имя параметра

Результат

Значение параметра

```
inline double GetDblParam(const char *szParam)
```

Получает параметр с плавающей точкой

Параметры

`szParam` – **[in]** имя параметра

Результат

Значение параметра

```
inline std::string GetStringParam(const char *szParam)
```

Получает строковый параметр

Параметры

`szParam` – **[in]** имя параметра

Результат

Значение параметра

```
inline bool GetBoolParam(const char *szParam)
```

Получает логический параметр

Параметры

`szParam` – **[in]** имя параметра

Результат

Значение параметра

```
inline void SetIntParam(const char *szParam, int nVal)
```

Задаёт целочисленный параметр

Параметры

- `szParam` – **[in]** имя параметра
- `nVal` – **[in]** Новое значение параметра

```
inline void SetDblParam(const char *szParam, double dVal)
```

Задаёт параметр с плавающей точкой

Параметры

- `szParam` – **[in]** имя параметра
- `dVal` – **[in]** Новое значение параметра

```
inline void SetStringParam(const char *szParam, const char *cVal)
```

Задаёт строковый параметр

Параметры

- `szParam` – **[in]** имя параметра

- **cVal** – **[in]** Новое значение параметра

```
inline void SetBoolParam(const char *szParam, bool bVal)
```

Задаёт логический параметр

Параметры

- **szParam** – **[in]** имя параметра
- **bVal** – **[in]** Новое значение параметра

```
inline void Write(const char *szFileName, const char *name_mapping_file = "")
```

Записывает модель в файл. Тип будет определен по расширению

Параметры

- **szFileName** – **[in]** путь к записываемому файлу
- **name_mapping_file** – **[in]** путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

```
inline void WriteMps(const char *szFileName, const char *name_mapping_file = "")
```

Записывает модель в файл в формате MPS

Параметры

- **szFileName** – **[in]** путь к записываемому файлу
- **name_mapping_file** – **[in]** путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

```
inline void WriteLp(const char *szFileName, const char *name_mapping_file = "")
```

Записывает модель в файл в формате LP

Параметры

- **szFileName** – **[in]** путь к записываемому файлу
- **name_mapping_file** – **[in]** путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

```
inline Variable AddVariable(double lb = 0, double ub = inf, double obj = 0.0,  
                           variable_type var_type = variable_type::continuous, const  
                           char *szName = "")
```

Добавляет переменную в модель с заданными параметрами

Параметры

- **lb** – **[in]** нижняя граница переменной
- **ub** – **[in]** верхняя граница переменной
- **obj** – **[in]** коэффициент к переменной в целевой функции
- **var_type** – **[in]** тип переменной
- **szName** – **[in]** имя переменной

Результат

объект новой переменной

```
inline Constraint AddConstraint(const LinearExpression &expr, constraint_sense sense,  
                               double rhs, const char *szName)
```

Добавляет ограничение в модель с заданными параметрами

Параметры

- **expr** – [in] линейное выражение для ограничения
- **sense** – [in] тип ограничения (\leq , \geq , $=$)
- **rhs** – [in] константное значение в правой части ограничения
- **szName** – [in] имя ограничения. Если передана пустая строка или nullptr, имя ограничения будет сгенерировано

Результат

объект нового ограничения

```
inline Constraint AddConstraint(const QuadExpression &expr, constraint_sense sense,  
                               double rhs, const char *szName)
```

Добавляет ограничение в модель с заданными параметрами

Параметры

- **expr** – [in] квадратичное выражение для ограничения
- **sense** – [in] тип ограничения (\leq , \geq , $=$)
- **rhs** – [in] константное значение в правой части ограничения
- **szName** – [in] имя ограничения. Если передана пустая строка или nullptr, имя ограничения будет сгенерировано

Результат

объект нового ограничения

```
inline Constraint AddConstraint(const LinearExpression &lhs, constraint_sense sense,  
                               const LinearExpression &rhs, const char *szName)
```

Добавляет ограничение в модель с заданными параметрами

Параметры

- **lhs** – [in] выражение для ограничения (левая часть)
- **sense** – [in] переменная знака ограничения (\leq , \geq , $=$)
- **rhs** – [in] выражение в правой части ограничения
- **szName** – [in] имя ограничения

Результат

объект нового ограничения

```
inline Constraint AddConstraint(const LinearExpression &expr, double lb, double ub,  
                               const char *szName)
```

Добавляет интервальное ограничение в модель с заданными параметрами, вида
 $lhs \leq expr \leq rhs$

Параметры

- **expr** – [in] выражение для ограничения
- **lb** – [in] нижняя граница ограничения
- **ub** – [in] верхняя граница ограничения
- **szName** – [in] имя ограничения

Результат

объект нового ограничения

```
inline Constraint AddConstraint(const QuadExpression &expr, double lb, double ub,  
                               const char *szName)
```

Добавляет интервальное ограничение в модель с заданными параметрами, вида
 $lhs \leq expr \leq rhs$

Параметры

- **expr** – [in] выражение для ограничения
- **lb** – [in] нижняя граница ограничения
- **ub** – [in] верхняя граница ограничения
- **szName** – [in] имя ограничения

Результат

объект нового ограничения

```
inline Constraint AddConstraint(const Constraint &constr, const char *szName)
```

Добавляет уже существующее ограничение в модель

Параметры

- **constr** – [in] ограничение
- **szName** – [in] имя ограничения

Результат

объект нового ограничения

```
inline Variable GetVariable(int idx) const
```

Получает переменную модели по индексу

Параметры

idx – [in] индекс переменной

Результат

объект переменной

```
inline int GetVariablesCount() const
```

Получает количество переменных в модели

Результат

количество переменных в модели

```
inline Variable GetVariableByName(const char *name)
```

Получает переменную из модели по имени

Параметры

name – [in] имя переменной

Результат

объект переменной

```
inline Constraint GetConstraint(int idx) const
```

Получает ограничение из модели по индексу

Параметры

`idx` – **[in]** индекс ограничения

Результат

объект ограничения

```
inline int GetConstraintsCount() const
```

Получает количество ограничений модели

Результат

количество ограничений модели

```
inline Constraint GetConstrByName(const char *szName)
```

Получает ограничение из модели по имени

Параметры

`szName` – **[in]** имя ограничения

Результат

объект ограничения

```
inline void SetObjectiveSense(objective_sense sense)
```

Задаёт тип оптимизации целевой функции - максимизация или минимизация

Параметры

`sense` – **[in]** тип оптимизации

```
inline objective_sense GetObjectiveSense() const
```

Получает тип оптимизации целевой функции

Результат

тип оптимизации

```
inline void SetObjectiveOffset(double constant)
```

Задаёт константу в выражении целевой функции

Параметры

`constant` – **[in]** значение константы в целевой функции

```
inline void SetObjective(const LinearExpression &expr, objective_sense sense =  
                        objective_sense::minimize)
```

Задаёт выражение целевой функции и тип оптимизации

Параметры

- `expr` – **[in]** линейное выражение
- `sense` – **[in]** тип оптимизации

```
inline void SetObjective(const QuadExpression &expr, objective_sense sense =  
                        objective_sense::minimize)
```

Задаёт выражение целевой функции и тип оптимизации

Параметры

- **expr** – [in] квадратичное выражение
- **sense** – [in] тип оптимизации

inline *LinearExpression* GetObjective() const

Получает выражение целевой функции

Результат

Объект выражения

inline *SolveResult* Solve()

Стартует оптимизацию модели

Результат

Объект с результатом оптимизации

inline *SolveResult* SolveRemote(const char *name_mapping_file)

Стартует оптимизацию модели на удаленном сервере в синхронном режиме. Предварительно необходимо установить переменную окружения X_API_KEY.

Параметры

name_mapping_file – [in] путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных

Результат

Объект с результатом оптимизации

inline void SolveRemoteAsync(const char *name_mapping_file)

Стартует оптимизацию модели на удаленном сервере в асинхронном режиме без ожидания результата. Предварительно необходимо установить переменную окружения X_API_KEY.

Параметры

name_mapping_file – [in] путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных

inline const char *GetRemoteSolveLog(const char *szCalcUID) const

Получает лог удалённого расчета

Параметры

szCalcUID – [in] идентификатор удалённого расчета

Результат

лог удалённого расчета

inline void Remove(*Variable* &var)

Удаляет переменную из модели

Параметры

var – [in] переменная

inline void Remove(*Constraint* &constr)

Удаляет ограничение из модели

Параметры

constr – [in] ограничение

`inline void Clear()`

Очищает модель от всех данных. Все старые переменные и ограничения становятся невалидными

`inline void SetLogFile(const char *szLogFile)`

Задать файл для записи лога решения

Параметры

`szLogFile` – **[in]** файл лога

`inline void SetLogCallback(ILogCallback *pcb)`

Задать интерфейс для обратного вызова при записи лога

Параметры

`pcb` – **[in]** интерфейс для обратного вызова

`inline void AddMipStartValue(const char *var_name, double var_value)`

Добавить возможное значение переменной в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

Параметры

- `var_name` – **[in]** имя переменной
- `var_value` – **[in]** значение переменной

`inline void AddMipStartValue(const Variable &var, double var_value)`

Добавить возможное значение переменной в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

Параметры

- `var` – **[in]** переменная
- `var_value` – **[in]** значение переменной

`inline void AddMipStartValues(const char *sol_file)`

Добавить возможные значения переменных в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

Параметры

`sol_file` – **[in]** путь к файлу с решением

`inline void ClearMipStartValues()`

Удалить все стартовые значения для поиска решения

`inline std::string GetName() const`

Получает имя модели

Результат

имя модели

`inline void SetName(const char *szName)`

Задаёт имя модели

Параметры

`szName` – **[in]** имя модели

inline std::string **GetCalcUID()** const

Получить уникальный идентификатор последнего удаленного расчета.

Результат

идентификатор удалённого расчета

inline bool **IsMip()** const

Проверяет является ли задача целочисленной.

Результат

true если задача целочисленная, иначе false.

inline bool **IsNonlinear()** const

Проверяет является ли задача нелинейной.

Результат

true если задача нелинейная, иначе false.

class **SolveResult**

Класс для работы с итогами расчета

Предоставляет доступ к информации о расчете (количество итераций, статус, значение целевой функции и пр.), а также к значениям переменных

Public Functions

inline *solve_result* **GetSolveResult()** const

Получает статус процесса расчетов

Результат

Значение статуса

inline *solution_status* **GetSolutionStatus()** const

Получает статус модели в итоге расчета

Результат

Значение статуса

inline double **GetRelativeGap()** const

Получает точность решения в процентах

Результат

точность решения в процентах

inline unsigned int **GetIterationsCount()** const

Получает количество итераций, проведенных в процессе расчета

Результат

Кол-во итераций

inline std::int64_t **GetProcessedNodesCount()** const

Получает количество обработанных узлов дерева решений

Результат

Кол-во обработанных узлов

`inline double GetObjectiveFunctionValue() const`

Получает значение целевой функции

Результат

Значение целевой функции

`inline double GetBestBoundValue() const`

Получает значение граничной (двойственной) функции

Результат

Значение граничной (двойственной) функции

`inline double GetSolveTime() const`

Получает общее время решения

Результат

Общее время решения, секунды

`inline double GetVariableValue(const Variable &var) const`

Получает значение переменной в решении

Параметры

`var` – **[in]** переменная

Результат

Значение переменной в решении

`inline double GetVariableValue(const char *var_name) const`

Получает значение переменной в решении по имени

Параметры

`var_name` – **[in]** имя переменной

Результат

Значение переменной в решении

`inline double GetDualValue(const Constraint &constr) const`

Получает значение dual value

Параметры

`constr` – **[in]** ограничение

Результат

Значение dual value

`inline double GetDualValue(const char *constr_name) const`

Получает значение dual value по имени ограничения

Параметры

`constr_name` – **[in]** имя ограничения

Результат

Значение dual value

`inline double GetReducedCost(const Variable &var) const`

Получает значение reduced cost

Параметры

`var` – **[in]** переменная

Результат

Значение reduced cost

```
inline double GetReducedCost(const char *var_name) const
```

Получает значение reduced cost по имени переменной

Параметры`var_name` – [in] имя переменной**Результат**

Значение reduced cost

```
inline double GetExpressionValue(const LinearExpression &expr) const
```

Получает значение выражения

Параметры`expr` – [in] линейное выражение**Результат**

Значение выражения

```
inline double GetExpressionValue(const QuadExpression &expr) const
```

Получает значение выражения

Параметры`expr` – [in] квадратичное выражение**Результат**

Значение выражения

```
inline void WriteSolution(const char *file_name) const
```

Записывает решение в файл

Параметры`file_name` – [in] путь к файлу решения для записи

```
class Variable
```

Переменная

Работа с переменной - чтение/изменение имени, верхней и нижней границы, типа, а также удаление

Public Functions

```
inline const char *GetName() const
```

Получает имя переменной в модели

Результат

имя переменной

```
inline void SetName(const char *szName)
```

Задаёт имя переменной в модели

Параметры`szName` – [in]

```
inline arhiplex::variable_type GetType() const
```

Получает тип переменной

Результат

тип переменной

```
inline void SetType(arhiplex::variable_type type)
```

Задаёт тип переменной в модели

Параметры

type – **[in]** тип переменной

```
inline void SetUpperBound(double upper_bound)
```

Задаёт верхнюю границу для переменной

Параметры

upper_bound – **[in]** новая верхняя граница переменной

```
inline double GetUpperBound() const
```

Получает верхнюю границу переменной

Результат

верхняя граница переменной

```
inline void SetLowerBound(double lower_bound)
```

Задаёт нижнюю границу для переменной

Параметры

lower_bound – **[in]** новая нижняя граница переменной

```
inline double GetLowerBound() const
```

Получает нижнюю границу переменной

Результат

нижняя граница переменной

```
inline void Remove()
```

Помечает переменную как удаленную. Переменная не будет участвовать в расчетах

```
enum class arhiplex::variable_type
```

Тип переменной

Values:

```
enumerator continuous
```

непрерывная

```
enumerator binary
```

бинарная

```
enumerator integer
```

целочисленная

enumerator **semicontinuous**

полу-непрерывная

enumerator **semiinteger**

полу-целая

enum class **arhiplex::objective_sense**

Направление оптимизации целевой функции

Values:

enumerator **minimize**

Минимизация

enumerator **maximize**

Максимизация

enum class **arhiplex::constraint_sense**

Знак ограничения между левой частью (выражением) и правой частью (константой)

Values:

enumerator **equal**

„=“

enumerator **less_equal**

„≤“

enumerator **greater_equal**

„≥“

enum class **arhiplex::solve_result**

Результат процесса расчетов

Values:

enumerator **success**

Процесс решения успешен - найдено некоторое решение или обнаружена недо-
стижимость модели

enumerator **fail**

Процесс решения неудачен - не найдено никаких решений и не обнаружена недо-
стижимость модели

enumerator **remote_invalid_api_key**

Невалидный ключ для удаленного расчета

enumerator **remote_api_key_not_set**

Переменная окружения X_API_KEY не установлена

enumerator **remote_time_amount_is_over**

Не осталось доступного времени для осуществления удаленного расчета

enumerator **remote_time_per_calc_violated**

Превышен лимит времени для единичного удаленного расчета. Параметр `time_limit` нарушает ограничения лицензии

enumerator **remote_fail**

Удаленный расчет неудачен

enum class **arhiplex::solution_status**

Статус решения по результатам расчета

Values:

enumerator **invalid_solution**

неопределенное/невалидное значение

enumerator **optimal**

решение оптимально (погрешность в рамках заданного значения)

enumerator **feasible**

решение найдено, но не оптимально (погрешность > заданного значения)

enumerator **infeasible**

модель не имеет решения (решение недостижимо)

enumerator **unbounded**

модель неограничена (целевая функция может бесконечно неограниченно увеличиваться/уменьшаться)

enumerator **infeasible_or_unbounded**

модель недостижима или неограничена

`arhiplex.ArhiplexException : Exception`

Исключение, генерируемое всеми классами в случае ошибки

`arhiplex.Constraint : IDisposable`

Работа с ограничениями у модели, позволяет работать с выражением и границами

Public Functions

`string? GetName ()`

Получает имя ограничения

Результат

имя ограничения

`void SetName(string name)`

Задаёт имя ограничения в модели

Параметры

`name` – имя ограничение

`void Remove()`

Помечает ограничение как удаленное. Такое ограничение будет исключено из расчетов

`LinearExpression GetLinearExpression()`

Линейное выражение, связанное с ограничением

Результат

Линейное выражение

QuadExpression GetQuadExpression()

Квадратичное выражение, связанное с ограничением

Результат

Квадратичное выражение

void SetUpperBound(double upper_bound)

Задаёт верхнюю границу для ограничения

Параметры

`lower_bound` – новая верхняя граница ограничения

double GetUpperBound()

Получает верхнюю границу ограничения

Результат

верхняя граница ограничения

void SetLowerBound(double lower_bound)

Задаёт нижнюю границу для ограничения

Параметры

`lower_bound` – новая нижняя граница ограничения

double GetLowerBound()

Получает нижнюю границу ограничения

Результат

нижняя граница ограничения

double GetRange()

Получает интервал ограничения: `upper_bound - lower_bound`

Результат

`upper_bound - lower_bound`

void SetRange(double range)

Задаёт интервал ограничения: `lower_bound <= expression <= lower_bound + range`

Параметры

`range` – интервал для ограничения

arhiplex.LinearExpression : IDisposable

Обеспечивает работу с линейными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная * коэффициент.

Public Functions

`LinearExpression(double offset)`

Создает пустое выражение с возможностью задать константу

Параметры

`offset` – константа в выражении

`LinearExpression(Variable var, double coeff = 1.0)`

Создает выражение на основе переменной и коэффициента

Параметры

- `var` – переменная в выражении
- `coeff` – коэффициент переменной в выражении

`int GetTermsCount()`

Возвращает кол-во элементов в выражении

Результат

кол-во элементов в выражении

`Variable GetTermVariable(int term_idx)`

Возвращает переменную по индексу элемента в выражении

Параметры

`term_idx` – индекс элемента в выражении

Результат

Объект переменной

`double GetTermCoeff(int term_idx)`

Возвращает коэффициент переменной по индексу элемента в выражении

Параметры

`term_idx` – индекс элемента в выражении

Результат

Значение коэффициента переменной

`void SetTermCoeff(int term_idx, double value)`

Задаёт коэффициент переменной по индексу выражения

Параметры

- `term_idx` – индекс элемента в выражении
- `value` – Значение коэффициента при переменной в элементе

`double GetConstant()`

Возвращает константу в выражении

Результат

значение константы

`void SetConstant(double constant)`

Задаёт константу в выражении

Параметры

constant – значение константы

void AddConstant(double constant)

Добавляет константу к выражению

Параметры

constant – значение константы

void AddTerm(Variable var, double coeff)

Добавляет элемент к выражению

Параметры

- **var** – переменная
- **coeff** – коэффициент к переменной

void AddExpression(*LinearExpression* expr, double mult = 1.0)

Добавляет выражение к выражению

Параметры

- **expr** – добавляемое выражение
- **mult** – коэффициент к добавляемому выражению

void RemoveTerm(int term_idx)

Удаляет элемент выражения по индексу

Параметры

term_idx – индекс удаляемого элемента в выражении

void RemoveVariable(Variable var)

Удаляет переменную из выражения

Параметры

var – удаляемая переменная

QuadExpression GetQuadPart()

Возвращает квадратичную часть выражения

Результат

квадратичная часть выражения

LinearExpression CreateFreeCopy()

Создает копию выражения, не привязанную к модели или другому ограничению

Результат

копия выражения

arhiplex.QuadExpression : IDisposable

Обеспечивает работу с квадратичными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная* переменная * коэффициент.

Public Functions

`QuadExpression()`

Создает пустое выражение

`QuadExpression(Variable var1, Variable var2, double coeff = 1.0)`

Создает выражение на основе переменных и коэффициента

Параметры

- `var1` – переменная №1 в выражении
- `var2` – переменная №2 в выражении
- `coeff` – коэффициент в выражении

`int GetTermsCount()`

Возвращает кол-во элементов в выражении

Результат

Кол-во элементов в выражении

`Variable GetTermVariable1(int term_idx)`

Возвращает переменную №1 по индексу элемента в выражении

Параметры

`term_idx` – индекс элемента в выражении

Результат

объект переменной

`Variable GetTermVariable2(int term_idx)`

Возвращает переменную №2 по индексу элемента в выражении

Параметры

`term_idx` – индекс элемента в выражении

Результат

объект переменной

`double GetTermCoeff(int term_idx)`

Возвращает коэффициент переменных по индексу элемента в выражении

Параметры

`term_idx` – индекс элемента в выражении

Результат

Значение коэффициента

`void SetTermCoeff(int term_idx, double value)`

Задаёт коэффициент переменной по индексу выражения

Параметры

- `term_idx` – индекс элемента в выражении
- `value` – Значение коэффициента при переменных в элементе

`void AddTerm(Variable var1, Variable var2, double coeff)`

Добавляет элемент к выражению

Параметры

- `var1` – переменная
- `var2` – переменная
- `coeff` – коэффициент

`void AddExpression(QuadExpression quad_expr, double mult = 1.0)`

Добавляет выражение к выражению

Параметры

- `quad_expr` – квадратичное выражение
- `coeff` – коэффициент к добавляемому выражению

`void RemoveTerm(int term_idx)`

Удаляет элемент выражения по индексу

Параметры

`term_idx` – индекс удаляемого элемента в выражении

LinearExpression `GetLinearPart()`

Возвращает линейную часть выражения

Результат

линейная часть

QuadExpression `CreateFreeCopy()`

Создает копию выражения, не привязанную к модели или другому ограничению

Результат

копия выражения

`arhiplex.Model : IDisposable`

Предоставляет функции чтения/записи файлов, модификации модели, управления переменными, ограничениями, а также целевой функцией

Public Functions

`Model(string name = "")`

Создает экземпляр модели управления переменными, ограничениями, а также целевой функцией

Параметры

`name` – Имя модели

`void Read(string file_name)`

Читает модель из файла, тип модели определяется по расширению

Параметры

`file_name` – путь к файлу модели

`void ReadMps(string file_name)`

Читает модель из файла, при этом он будет читаться как MPS файл независимо от расширения

Параметры

`file_name` – путь к файлу модели

`void ReadLp(string file_name)`

Читает модель из файла, при этом он будет читаться как LP файл независимо от расширения

Параметры

`file_name` – путь к файлу модели

`void Write(string file_name, string name_mapping_file = "")`

Записывает модель в файл. Тип будет определен по расширению

Параметры

- `file_name` – путь к записываемому файлу
- `name_mapping_file` – путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`void WriteMps(string file_name, string name_mapping_file = "")`

Записывает модель в файл в формате MPS

Параметры

- `file_name` – путь к записываемому файлу
- `name_mapping_file` – путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`void WriteLp(string file_name, string name_mapping_file = "")`

Записывает модель в файл в формате LP

Параметры

- `file_name` – путь к записываемому файлу
- `name_mapping_file` – путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`int GetIntParam(string param_name)`

Получает целочисленный параметр

Параметры

`param_name` – имя параметра

Результат

Значение параметра

`double GetDblParam(string param_name)`

Получает параметр с плавающей точкой

Параметры

`param_name` – имя параметра

Результат

Значение параметра

`string GetStringParam(string param_name)`

Получает строковый параметр

Параметры

`param_name` – имя параметра

Результат

Значение параметра

`bool GetBoolParam(string param_name)`

Получает логический параметр

Параметры

`param_name` – имя параметра

Результат

Значение параметра

`void SetIntParam(string param_name, int nVal)`

Задаёт целочисленный параметр

Параметры

- `param_name` – имя параметра
- `nVal` – Новое значение параметра

`void SetDbiParam(string param_name, double dVal)`

Задаёт параметр с плавающей точкой

Параметры

- `param_name` – имя параметра
- `dVal` – Новое значение параметра

`void SetStringParam(string param_name, string cVal)`

Задаёт строковый параметр

Параметры

- `param_name` – имя параметра
- `cVal` – Новое значение параметра

`void SetBoolParam(string param_name, bool bVal)`

Задаёт логический параметр

Параметры

- `param_name` – имя параметра
- `bVal` – Новое значение параметра

`string? GetName ()`

Получает имя модели

Результат

Имя модели

`void SetName(string name)`

Задаёт имя модели

Параметры

`name` – Имя модели

`Variable AddVariable(double lb, double ub, double obj, Variable_type var_type, string name)`

Добавляет переменную в модель с заданными параметрами

Параметры

- `lb` – нижняя граница переменной
- `ub` – верхняя граница переменной
- `obj` – коэффициент к переменной в целевой функции
- `var_type` – тип переменной
- `name` – имя переменной

`Constraint AddConstraint(LinearExpression lin_expr, Constraint_sense sense, double rhs, string name)`

Добавляет ограничение в модель с заданными параметрами

Параметры

- `lin_expr` – линейное выражение для ограничения
- `sense` – тип ограничения (`<=`, `>=`, `==`)
- `rhs` – константное значение в правой части ограничения
- `name` – имя ограничения. Если передана пустая строка или `null`, имя ограничения будет сгенерировано

Результат

объект нового ограничения

`Constraint AddConstraint(QuadExpression quad_expr, Constraint_sense sense, double rhs, string name)`

Добавляет ограничение в модель с заданными параметрами

Параметры

- `quad_expr` – квадратичное выражение для ограничения
- `sense` – тип ограничения (`<=`, `>=`, `==`)
- `rhs` – константное значение в правой части ограничения
- `name` – имя ограничения. Если передана пустая строка или `null`, имя ограничения будет сгенерировано

Результат

объект нового ограничения

Constraint AddConstraint(*LinearExpression* lin_expr, double lb, double ub, string name)

Добавляет ограничение в модель с заданными параметрами

Параметры

- **lin_expr** – линейное выражение для ограничения
- **lb** – нижняя граница ограничения
- **ub** – верхняя граница ограничения
- **name** – имя ограничения. Если передана пустая строка или null, имя ограничения будет сгенерировано

Результат

объект нового ограничения

Constraint AddConstraint(*QuadExpression* quad_expr, double lb, double ub, string name)

Добавляет ограничение в модель с заданными параметрами

Параметры

- **quad_expr** – квадратичное выражение для ограничения
- **lb** – нижняя граница ограничения
- **ub** – верхняя граница ограничения
- **name** – имя ограничения. Если передана пустая строка или null, имя ограничения будет сгенерировано

Результат

объект нового ограничения

Constraint AddConstraint(Constraint constr, string name)

Добавляет уже существующее ограничение в модель

Параметры

- **constr** – ограничение
- **name** – имя ограничения

Результат

объект нового ограничения

int GetVariablesCount()

Получает количество переменных в модели

Результат

количество переменных в модели

Variable GetVariable(int var_idx)

Получает переменную модели по индексу

Параметры

var_idx – индекс переменной

Результат

объект переменной

Variable **GetVariable**(string name)

Получает переменную модели по имени

Параметры

name – имя переменной

Результат

объект переменной

int **GetConstraintsCount**()

Получает количество ограничений модели

Результат

количество ограничений модели

Constraint **GetConstraint**(int constr_idx)

Получает ограничение из модели по индексу

Параметры

constr_idx – индекс ограничения

Результат

объект ограничения

Constraint **GetConstraint**(string name)

Получает ограничение из модели по имени

Параметры

name – имя ограничения

Результат

объект ограничения

void **SetObjectiveSense**(Objective_sense sense)

Задаёт тип оптимизации целевой функции - максимизация или минимизация

Параметры

sense – тип оптимизации

Objective_sense **GetObjectiveSense**()

Получает тип оптимизации целевой функции

Результат

тип оптимизации целевой функции

void **SetObjectiveOffset**(double offset)

Задаёт константу в выражении целевой функции

Параметры

offset – значение константы в целевой функции

void **SetObjective**(*LinearExpression* expr, Objective_sense sense =
Objective_sense.minimize)

Задаёт выражение целевой функции и тип оптимизации

Параметры

- **expr** – линейное выражение
- **sense** – тип оптимизации

void **SetObjective**(*QuadExpression* quad_expr, Objective_sense sense = Objective_sense.minimize)

Задаёт выражение целевой функции и тип оптимизации

Параметры

- **expr** – квадратичное выражение
- **sense** – тип оптимизации

void **ClearObjective**()

Удаляет целевую функцию из модели

LinearExpression **GetObjective**()

Получает выражение целевой функции

Результат

Получает выражение целевой функции

QuadExpression **GetQuadObjective**()

Получает выражение целевой функции

Результат

Получает выражение целевой функции

SolveResult **Solve**()

Стартует оптимизацию модели

Результат

Объект с результатом оптимизации

SolveResult **SolveRemote**(string name_mapping_file)

Стартует оптимизацию модели на удаленном сервере в синхронном режиме. Предварительно необходимо установить переменную окружения X_API_KEY.

Параметры

name_mapping_file – путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных

Результат

Объект с результатом оптимизации

void **SolveRemoteAsync**(string name_mapping_file)

Стартует оптимизацию модели на удаленном сервере в асинхронном режиме без ожидания результата. Предварительно необходимо установить переменную окружения X_API_KEY.

Параметры

name_mapping_file – путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных

`string? GetRemoteSolveLog (string calc_uid)`

Получает лог удалённого расчёта

Параметры

`calc_uid` – идентификатор удалённого расчёта

Результат

лог удалённого расчёта

`void Remove (Variable var)`

Удаляет переменную из модели

Параметры

`var` – переменная

`void Remove (Constraint constr)`

Удаляет ограничение из модели

Параметры

`constr` – ограничение

`void Clear()`

Очищает модель от всех данных. Все старые переменные и ограничения становятся невалидными

`void SetLogFile (string log_file)`

Задать файл для записи лога решения

Параметры

`log_file` – файл лога

`void AddMipStartValue (string var_name, double var_value)`

Добавить возможное значение переменной в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

Параметры

- `var_name` – имя переменной
- `var_value` – значение переменной

`void AddMipStartValue (Variable var, double var_value)`

Добавить возможное значение переменной в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

Параметры

- `var` – переменная
- `var_value` – значение переменной

`void AddMipStartValues (string sol_file)`

Добавить возможные значения переменных в решении в качестве «подсказки». Стартовые значения очищаются после начала процесса решения.

Параметры

`sol_file` – путь к файлу с решением

`void ClearMipStartValues()`

Удалить все стартовые значения для поиска решения

`string? GetCalcUID ()`

Получить уникальный идентификатор последнего удаленного расчета.

Результат

идентификатор удаленного расчета.

`bool IsMip()`

Проверяет является ли задача целочисленной.

Результат

true если задача целочисленная, иначе false.

`bool IsNonlinear()`

Проверяет является ли задача целочисленной.

Результат

true если задача нелинейная, иначе false.

`arhiplex.SolveResult : IDisposable`

Предоставляет доступ к информации о расчете (количество итераций, статус, значение целевой функции и пр.), а также к значениям переменных

Public Functions

`Solve_result GetSolveResult()`

Получает статус процесса расчетов

Результат

Значение статуса

`Solution_status GetSolutionStatus()`

Получает статус модели в итоге расчета

Результат

Значение статуса

`double GetRelativeGap()`

Получает точность решения в процентах

Результат

точность решения в процентах

`uint GetIterationsCount()`

Получает количество итераций, проведенных в процессе расчета

Результат

Кол-во итераций

`Int64 GetProcessedNodesCount()`

Получает количество обработанных узлов дерева решений

Результат

Кол-во обработанных узлов

`double GetObjectiveFunctionValue()`

Получает значение целевой функции

Результат

Значение целевой функции

`double GetBestBoundValue()`

Получает значение граничной (двойственной) функции

Результат

Значение граничной (двойственной) функции

`double GetSolveTime()`

Получает общее время решения

Результат

Общее время решения, секунды

`double GetVariableValue(Variable var)`

Получает значение переменной в решении

Параметры`var` – объект переменной**Результат**

Значение переменной в решении

`double GetVariableValue(string var_name)`

Получает значение переменной в решении по имени

Параметры`var_name` – имя переменной**Результат**

Значение переменной в решении

`double GetDualValue(Constraint constr)`

Получает значение dual value

Параметры`constr` – ограничение**Результат**

Значение dual value

`double GetDualValue(string constr_name)`

Получает значение dual value по имени ограничения

Параметры`constr` – имя ограничения**Результат**

Значение dual value

`double GetReducedCost(Variable var)`

Получает значение reduced cost

Параметры

`var` – переменная

Результат

Значение reduced cost

`double GetReducedCost(string var_name)`

Получает значение reduced cost

Параметры

`var` – имя переменной

Результат

Значение reduced cost

`double GetExpressionValue(LinearExpression expression)`

Получает значение выражения

Параметры

`expression` – линейное выражение

Результат

Значение выражения

`double GetExpressionValue(QuadExpression expression)`

Получает значение выражения

Параметры

`expression` – квадратичное выражение

Результат

Значение выражения

`void WriteSolution(string file_name)`

Записывает решение в файл

Параметры

`file_name` – путь к файлу решения для записи

`arhiplex.Variable : IDisposable`

Класс для работа с переменной - чтение/изменение имени, верхней и нижней границы, типа, а также удаление

Public Functions

`void SetUpperBound(double upper_bound)`

Задаёт верхнюю границу для переменной

Параметры

`upper_bound` – новая верхняя граница переменной

`double GetUpperBound()`

Получает верхнюю границу переменной

Результат

верхняя граница переменной

`void SetLowerBound(double lower_bound)`

Задаёт нижнюю границу для переменной

Параметры

`lower_bound` – новая нижняя граница переменной

`double GetLowerBound()`

Получает нижнюю границу переменной

Результат

нижняя граница переменной

`void Remove()`

Помечает переменную как удаленную. Переменная не будет участвовать в расчетах

`Variable_type GetVarType()`

Получает тип переменной

Результат

тип переменной

`void SetType(Variable_type type)`

Задаёт тип переменной в модели

Параметры

`type` – тип переменной

`string? GetName ()`

Получает имя переменной

Результат

имя переменной

`void SetName(string name)`

Задаёт имя переменной в модели

Параметры

`name` – имя переменной

`arhiplex.Variable_type`

Тип переменной

Values:

enumerator `continuous`

enumerator `binary`

непрерывная

enumerator `integer`

бинарная

enumerator **semicontinuous**

целочисленная

enumerator **semiinteger**

полу-непрерывная

arhiplex.Objective_sense

Направление оптимизации целевой функции

Values:

enumerator **minimize**

enumerator **maximize**

Минимизация

arhiplex.Constraint_sense

Знак ограничения между левой частью (выражением) и правой частью (константой)

Values:

enumerator **equal**

enumerator **less_equal**

„ $=$ “

enumerator **greater_equal**

„ \leq “

arhiplex.Solve_result

Результат процесса расчетов

Values:

enumerator **success**

enumerator **fail**

Процесс решения успешен - найдено некоторое решение или обнаружена недо-
стижимость модели

enumerator **remote_invalid_api_key**

Процесс решения неудачен - не найдено никаких решений и не обнаружена недо-
стижимость модели

enumerator **remote_api_key_not_set**

Невалидный ключ для удаленного расчета

enumerator **remote_time_amount_is_over**

Переменная окружения X_API_KEY не установлена

enumerator **remote_time_per_calc_violated**

Не осталось доступного времени для осуществления удаленного расчета

enumerator **remote_fail**

Превышен лимит времени для единичного удаленного расчета. Параметр time_limit нарушает ограничения лицензии

arhiplex.Solution_status

Статус решения по результатам расчета

Values:

enumerator **invalid_solution**

enumerator **optimal**

неопределенное/невалидное значение

enumerator **feasible**

решение оптимально (погрешность в рамках заданного значения)

enumerator **infeasible**

решение найдено, но не оптимально (погрешность > заданного значения)

enumerator **unbounded**

модель не имеет решения (решение недостижимо)

enumerator **infeasible_or_unbounded**

модель неограничена (целевая функция может бесконечно неограниченно увеличиваться/уменьшаться)

модель недостижима или неограничена

exception arhiplexpy.ArhiplexException

class arhiplexpy.Constraint

Работа с линейными ограничениями у модели,

позволяет работать с выражением и границами, а также удалять его

`get_linear_expression(self: arhiplexpy.Constraint) → arhiplex::LinearExpression`

Линейная часть выражения, связанного с ограничением

return

линейная часть выражения, связанного с ограничением

`get_lower_bound(self: arhiplexpy.Constraint) → float`

Получает нижнюю границу ограничения

return

нижнюю границу ограничения

`get_name(self: arhiplexpy.Constraint) → str`

Получает имя ограничения

return

имя ограничения

`get_quad_expression(self: arhiplexpy.Constraint) → arhiplex::QuadExpression`

Квадратичная часть выражения, связанного с ограничением

return

квадратичная часть выражения, связанного с ограничением

`get_range(self: arhiplexpy.Constraint) → float`

Получает интервал ограничения: `upper_bound - lower_bound`

return

`upper_bound - lower_bound`

`get_upper_bound(self: arhiplexpy.Constraint) → float`

Получает верхнюю границу ограничения

return

верхнюю границу ограничения

`remove(self: arhiplexpy.Constraint) → None`

Помечает ограничение как удаленное. Такое ограничение будет исключено из расчетов

`set_lower_bound(self: arhiplexpy.Constraint, lower_bound: float) → None`

Задаёт нижнюю границу ограничения

param lower_bound

новая нижняя граница ограничения

`set_name(self: arhiplexpy.Constraint, constr_name: str) → None`

Задаёт имя ограничения

param constr_name

имя ограничения

`set_range(self: arhiplexpy.Constraint, range: float) → None`

Задаёт интервал ограничения: $\text{lower_bound} \leq \text{expression} \leq \text{lower_bound} + \text{range}$

param range

интервал для ограничения

`set_upper_bound(self: arhiplexpy.Constraint, upper_bound: float) → None`

Задаёт верхнюю границу ограничения

param upper_bound

новая верхняя граница ограничения

`class arhiplexpy.LinearExpression`

Обеспечивает работу с линейными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная * коэффициент.

`add_constant(self: arhiplexpy.LinearExpression, value: float) → None`

Добавляет константу к выражению

param

value значение константы

`add_expression(self: arhiplexpy.LinearExpression, expr: arhiplexpy.LinearExpression, mult: float) → None`

Добавляет выражение к выражению

param expr

добавляемое выражение

param mult

коэффициент к добавляемому выражению

`add_term(self: arhiplexpy.LinearExpression, var: arhiplexpy.Variable, coeff: float = 1.0) → None`

Добавляет элемент к выражению

param var
переменная

param coeff
коэффициент к переменной

`copy(self: arhiplexpy.LinearExpression) → arhiplexpy.LinearExpression`

Создает копию выражения, не привязанную к модели или другому ограничению

return
копия выражения

`empty(self: arhiplexpy.LinearExpression) → bool`

Возвращает истину, если выражение пустое

return
истина, если выражение пустое

`get_constant(self: arhiplexpy.LinearExpression) → float`

Возвращает константу в выражении

return
значение константы

`get_name(self: arhiplexpy.LinearExpression) → str`

Получает имя выражения

return
имя выражения

`get_quad_part(self: arhiplexpy.LinearExpression) → arhiplex::QuadExpression`

Возвращает квадратичную часть выражения

return
квадратичная часть выражения

`get_term_coeff(self: arhiplexpy.LinearExpression, ind: int) → float`

Возвращает коэффициент переменной по индексу элемента в выражении

param ind
индекс элемента в выражении

return
значение коэффициента переменной

`get_term_variable(self: arhiplexpy.LinearExpression, ind: int) → arhiplexpy.Variable`

Возвращает переменную по индексу элемента в выражении

param ind
индекс элемента в выражении

return
объект переменной

`get_terms_count(self: arhiplexpy.LinearExpression) → int`

Возвращает количество элементов в выражении

return
количество элементов в выражении

`remove_term(self: arhiplexpy.LinearExpression, idx: int) → None`

Удаляет элемент выражения по индексу

param idx

индекс удаляемого элемента в выражении

`remove_variable(self: arhiplexpy.LinearExpression, var: arhiplexpy.Variable) → None`

Удаляет переменную из выражения

param var

удаляемая переменная

`set_constant(self: arhiplexpy.LinearExpression, value: float) → None`

Задаёт константу в выражении

param value

значение константы

`set_name(self: arhiplexpy.LinearExpression, expr_name: str) → None`

Задаёт имя выражения

param expr_name

имя выражения

`set_term_coeff(self: arhiplexpy.LinearExpression, ind: int, value: float) → None`

Задаёт коэффициент переменной по индексу выражения

param ind

индекс элемента в выражении

param value

значение коэффициента при переменной в элементе

`class arhiplexpy.Model`

Создает экземпляр модели

`add_constraint(*args, **kwargs)`

Overloaded function.

1. `add_constraint(self: arhiplexpy.Model, expr: arhiplexpy.LinearExpression, sense: arhiplexpy.constraint_sense, rhs: float, constr_name: str) -> arhiplexpy.Constraint`

Добавляет ограничение в модель с заданными параметрами

param expr

выражение для ограничения

param sense

тип ограничения (`<=`, `>=`, `==`)

param rhs

константное значение в правой части ограничения

param constr_name

имя ограничения

return

объект нового ограничения

2. `add_constraint(self: arhiplexpy.Model, expr: arhiplexpy.QuadExpression, sense: arhiplexpy.constraint_sense, rhs: float, constr_name: str) -> arhiplexpy.Constraint`

Добавляет ограничение в модель с заданными параметрами

param expr
выражение для ограничения

param sense
тип ограничения (`<=`, `>=` , `==`)

param rhs
константное значение в правой части ограничения

param constr_name
имя ограничения

return
объект нового ограничения

3. `add_constraint(self: arhiplexpy.Model, lhs: arhiplexpy.LinearExpression, sense: arhiplexpy.constraint_sense, rhs: arhiplexpy.LinearExpression, constr_name: str) -> arhiplexpy.Constraint`

Добавляет ограничение в модель с заданными параметрами

param lhs
выражение для ограничения (левая часть)

param sense
переменная знака ограничения (`<=`, `>=` , `==`)

param rhs
константное значение в правой части ограничения

param constr_name
имя ограничения

return
объект нового ограничения

4. `add_constraint(self: arhiplexpy.Model, expr: arhiplexpy.LinearExpression, lower_bound: float, upper_bound: float, constr_name: str) -> arhiplexpy.Constraint`

Добавляет интервальное ограничение в модель с заданными параметрами, вида `lhs <= expr <= rhs`

param expr
выражение для ограничения

param lower_bound
нижняя граница ограничения

param upper_bound
верхняя граница ограничения

param constr_name

имя ограничения

return

объект нового ограничения

5. `add_constraint(self: arhiplexpy.Model, expr: arhiplexpy.QuadExpression, lower_bound: float, upper_bound: float, constr_name: str) -> arhiplexpy.Constraint`

Добавляет интервальное ограничение в модель с заданными параметрами, вида $lhs \leq expr \leq rhs$

param expr

выражение для ограничения

param lower_bound

нижняя граница ограничения

param upper_bound

верхняя граница ограничения

param constr_name

имя ограничения

return

объект нового ограничения

6. `add_constraint(self: arhiplexpy.Model, constr: arhiplexpy.Constraint, constr_name: str) -> arhiplexpy.Constraint`

Добавляет уже существующее ограничение в модель

param constr

выражение для ограничения

param constr_name

имя ограничения

return

объект нового ограничения

`add_constraints(self: arhiplexpy.Model, constraints: dict) → None`

Добавляет ограничения в модель

param constraints

объект dictionary(name -> constraint)

`add_mip_start_values(*args, **kwargs)`

Overloaded function.

1. `add_mip_start_values(self: arhiplexpy.Model, start_values: dict) -> None`

Задать начальные значения переменных в решении в качестве «подсказки». Начальные значения очищаются после начала процесса решения.

param start_values

объект словаря (имя : значение)

2. `add_mip_start_values(self: arhiplexpy.Model, sol_file: str) -> None`

Задать начальные значения переменных в решении, указав файл решения

param sol_file
путь к файлу решения

`add_variable(self: arhiplexpy.Model, lower_bound: float = 0.0, upper_bound: float = inf, obj_value: float = 0.0, var_type: arhiplexpy.variable_type = <variable_type.continuous: 0>, var_name: str = '') -> arhiplexpy.Variable`

Добавляет переменную в модель с заданными параметрами

param lower_bound
нижняя граница переменной

param upper_bound
верхняя граница переменной

param obj_value
коэффициент к переменной в целевой функции

param var_type
тип переменной

param var_name
имя переменной

return
объект новой переменной

`binary_var_list(self: arhiplexpy.Model, indicies: list, name: str) -> list`

Добавляет лист бинарных переменных

param indicies
список индексов, добавляемых к начальному имени (или список кортежей)

param name
префикс имени каждой добавленной переменной

return
список переменных

`clear(self: arhiplexpy.Model) -> None`

Очищает модель от всех данных. Все старые переменные и ограничения становятся невалидными

`clear_mip_start_values(self: arhiplexpy.Model) -> None`

Удалить все стартовые значения для поиска решения

`continuous_var_list(*args, **kwargs)`

Overloaded function.

1. `continuous_var_list(self: arhiplexpy.Model, indicies: list, lb: list, ub: list, name: str) -> list`

Добавляет лист непрерывных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

список нижних границ

param ub

список верхних границ

param name

префикс имени каждой добавленной переменной

return

список переменных

2. `continuous_var_list(self: arhiplexpy.Model, indices: list, lb: float = 0.0, ub: float = inf, name: str) -> list`

Добавляет лист непрерывных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

значение нижней границы (для всех переменных)

param ub

значение верхней границы (для всех переменных)

param name

префикс имени каждой добавленной переменной

return

список переменных

3. `continuous_var_list(self: arhiplexpy.Model, indices: list, lb: float = 0.0, ub: list, name: str) -> list`

Добавляет лист непрерывных переменных

param indices

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

значение нижней границы (для всех переменных)

param ub

список верхних границ

param name

префикс имени каждой добавленной переменной

return

список переменных

4. `continuous_var_list(self: arhiplexpy.Model, indicies: list, lb: list, ub: float = inf, name: str) -> list`

Добавляет лист непрерывных переменных

param indicies

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

список нижних границ

param ub

значение верхней границы (для всех переменных)

param name

префикс имени каждой добавленной переменной

return

список переменных

`get_bool_param(self: arhiplexpy.Model, param_name: str) -> bool`

Получает логический параметр

param param_name

имя параметра

return

значение параметра

`get_calc_uid(self: arhiplexpy.Model) -> str`

Получить уникальный идентификатор последнего удаленного расчета. :return: строку с уникальным идентификатором

`get_constraint(*args, **kwargs)`

Overloaded function.

1. `get_constraint(self: arhiplexpy.Model, idx: int) -> arhiplexpy.Constraint`

Получает ограничение из модели по индексу

param idx

индекс ограничения

return

объект ограничения

2. `get_constraint(self: arhiplexpy.Model, constr_name: str) -> arhiplexpy.Constraint`

Получает ограничение из модели по имени

param constr_name

имя ограничения

return

объект ограничения

`get_constraints_count(self: arhiplexpy.Model) → int`

Получает количество ограничений в модели :return: количество ограничений в модели

`get_dbl_param(self: arhiplexpy.Model, param_name: str) → float`

Получает параметр с плавающей точкой

param param_name

имя параметра

return

значение параметра

`get_int_param(self: arhiplexpy.Model, param_name: str) → int`

Получает целочисленный параметр

param param_name

имя параметра

return

значение параметра

`get_name(self: arhiplexpy.Model) → str`

Получает имя модели

return

имя модели

`get_objective(self: arhiplexpy.Model) → arhiplexpy.LinearExpression`

Получает выражение целевой функции

return

объект выражения

`get_objective_sense(self: arhiplexpy.Model) → arhiplexpy.objective_sense`

Получает тип оптимизации целевой функции

return

тип оптимизации

`get_remote_solve_log(self: arhiplexpy.Model, arg0: str) → str`

Получает лог удалённого расчета

:param calc_uid : идентификатор удалённого расчета :return: лог удалённого расчета

`get_string_param(self: arhiplexpy.Model, param_name: str) → str`

Получает строковый параметр

param param_name

имя параметра

return

значение параметра

`get_variable(*args, **kwargs)`

Overloaded function.

1. `get_variable(self: arhiplexpy.Model, idx: int) -> arhiplexpy.Variable`

Получает переменную модели по индексу

param idx
индекс переменной
return
объект переменной

2. `get_variable(self: arhiplexpy.Model, var_name: str) -> arhiplexpy.Variable`

Получает переменную из модели по имени

param var_name
имя переменной
return
объект переменной

`get_variables_count(self: arhiplexpy.Model) -> int`

Получает количество переменных в модели

return
количество переменных в модели

`integer_var_list(*args, **kwargs)`

Overloaded function.

1. `integer_var_list(self: arhiplexpy.Model, indices: list, lb: list, ub: list, name: str) -> list`

Добавляет лист целочисленных переменных

param indices
список индексов, добавляемых к начальному имени (или список кортежей)
param lb
список нижних границ
param ub
список верхних границ
param name
префикс имени каждой добавленной переменной
return
список переменных

2. `integer_var_list(self: arhiplexpy.Model, indices: list, lb: float = 0.0, ub: float = inf, name: str) -> list`

Добавляет лист целочисленных переменных

param indices
список индексов, добавляемых к начальному имени (или список кортежей)
param lb
значение нижней границы (для всех переменных)

param ub

значение верхней границы (для всех переменных)

param name

префикс имени каждой добавленной переменной

return

список переменных

3. `integer_var_list(self: arhiplexpy.Model, indicies: list, lb: float = 0.0, ub: list, name: str) -> list`

Добавляет лист целочисленных переменных

param indicies

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

значение нижней границы (для всех переменных)

param ub

список верхних границ

param name

префикс имени каждой добавленной переменной

return

список переменных

4. `integer_var_list(self: arhiplexpy.Model, indicies: list, lb: list, ub: float = inf, name: str) -> list`

Добавляет лист целочисленных переменных

param indicies

список индексов, добавляемых к начальному имени (или список кортежей)

param lb

список нижних границ

param ub

значение верхней границы (для всех переменных)

param name

префикс имени каждой добавленной переменной

return

список переменных

`is_mip(self: arhiplexpy.Model) -> bool`

Проверяет является ли задача целочисленной

`is_nonlinear(self: arhiplexpy.Model) -> bool`

Проверяет является ли задача нелинейной

`read(self: arhiplexpy.Model, file_name: str) → None`

Читает модель из файла

param file_name

имя файла

`read_lp(self: arhiplexpy.Model, file_name: str) → None`

Читает модель из файла, при этом он будет читаться как LP файл независимо от расширения

param file_name

путь к файлу модели

`read_mps(self: arhiplexpy.Model, file_name: str) → None`

Читает модель из файла, при этом он будет читаться как MPS файл независимо от расширения

param file_name

путь к файлу модели

`remove(*args, **kwargs)`

Overloaded function.

1. `remove(self: arhiplexpy.Model, constr: arhiplexpy.Constraint) -> None`

Удаляет ограничение из модели

param constr

ограничение

2. `remove(self: arhiplexpy.Model, var: arhiplexpy.Variable) -> None`

Удаляет переменную из модели

param var

переменная

`set_bool_param(self: arhiplexpy.Model, param_name: str, param_value: bool) → None`

Задаёт логический параметр

param param_value

значение параметра

`set_dbl_param(self: arhiplexpy.Model, param_name: str, param_value: float) → None`

Задаёт параметр с плавающей точкой

param param_name

имя параметра

param param_value

значение параметра

`set_int_param(self: arhiplexpy.Model, param_name: str, param_value: int) → None`

Задаёт целочисленный параметр

param param_name

имя параметра

param param_value

значение параметра

set_log_callback(*self*: arhiplexpy.Model, *callback*: object) → None

Задаёт функтор обратного вызова для вывода сообщений от солвера

param callback

функтор обратного вызова

set_log_file(*self*: arhiplexpy.Model, *log_file*: str) → None

Задать файл для записи лога решения

:param log_file файл лога

set_name(*self*: arhiplexpy.Model, *expr_name*: str) → None

Задаёт имя модели

param expr_name

имя модели

set_objective(*args, **kwargs)

Overloaded function.

1. **set_objective**(*self*: arhiplexpy.Model, *expr*: arhiplexpy.LinearExpression, *sense*: arhiplexpy.objective_sense) -> None

Задаёт выражение целевой функции и тип оптимизации

param expr

выражение

param sense

тип оптимизации

2. **set_objective**(*self*: arhiplexpy.Model, *expr*: arhiplexpy.QuadExpression, *sense*: arhiplexpy.objective_sense) -> None

Задаёт выражение целевой функции и тип оптимизации

param expr

выражение

param sense

тип оптимизации

set_objective_offset(*self*: arhiplexpy.Model, *offset*: float) → None

Задаёт константу в выражении целевой функции

param offset

значение константы

set_objective_sense(*self*: arhiplexpy.Model, *constr_name*: arhiplexpy.objective_sense) → None

Задаёт тип оптимизации целевой функции - максимизация или минимизация

param sense

тип оптимизации

`set_string_param(self: arhiplexpy.Model, param_name: str, param_value: str) → None`

Задаёт строковый параметр

param param_value
значение параметра

`solve(self: arhiplexpy.Model) → arhiplexpy.SolveResult`

Стартует оптимизацию модели

return
объект с результатом оптимизации

`solve_remote(self: arhiplexpy.Model, name_mapping_file: str) → arhiplexpy.SolveResult`

Стартует оптимизацию модели на удаленном сервере в синхронном режиме. Предварительно необходимо установить переменную окружения X_API_KEY.

param name_mapping_file путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных :**return** объект с результатом оптимизации

`solve_remote_async(self: arhiplexpy.Model, name_mapping_file: str) → None`

Стартует оптимизацию модели на удаленном сервере в асинхронном режиме без ожидания результата. Предварительно необходимо установить переменную окружения X_API_KEY.n

:param name_mapping_file путь к файлу, который будет записан и будет содержать соответствие оригинальных имен модели и анонимизированных

`sum(self: arhiplexpy.Model, arg0: list) → arhiplexpy.LinearExpression`

Суммирует лист из переменных и/или линейных выражений

return
объект выражения

`write(self: arhiplexpy.Model, file_name: str, name_mapping_file: str = '') → None`

Записывает модель в файл. Тип будет определен по расширению

param file_name
путь к записываемому файлу

:param name_mapping_file путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`write_lp(self: arhiplexpy.Model, file_name: str, name_mapping_file: str = '') → None`

Записывает модель в файл в формате LP

param file_name
путь к записываемому файлу

:param name_mapping_file путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`write_mps(self: arhiplexpy.Model, file_name: str, name_mapping_file: str = '') → None`

Записывает модель в файл в формате MPS

param file_name

путь к записываемому файлу

:param name_mapping_file путь к файлу, который будет содержать соответствие между именами модели при анонимизации (если пустой - запись без анонимизации)

`class arhiplexpy.QuadExpression`

Обеспечивает работу с нелинейными выражениями - добавление/удаление элементов и изменение коэффициентов. Элемент выражения - переменная * переменная * коэффициент.

`add_expression(*args, **kwargs)`

Overloaded function.

1. `add_expression(self: arhiplexpy.QuadExpression, expr: arhiplexpy.QuadExpression, mult: float) -> None`

Добавляет выражение к выражению

param expr

добавляемое выражение

param mult

коэффициент к добавляемому выражению

2. `add_expression(self: arhiplexpy.QuadExpression, expr: arhiplexpy.LinearExpression, mult: float) -> None`

Добавляет выражение к выражению

param expr

добавляемое выражение

param mult

коэффициент к добавляемому выражению

`add_term(self: arhiplexpy.QuadExpression, var1: arhiplexpy.Variable, var2: arhiplexpy.Variable, coeff: float = 1.0) → None`

Добавляет элемент к выражению

param var1

переменная 1

param var2

переменная 2

param coeff

коэффициент

`copy(self: arhiplexpy.QuadExpression) → arhiplexpy.QuadExpression`

Создает копию выражения, не привязанную к модели или другому ограничению

return

копия выражения

`empty(self: arhiplexpy.QuadExpression) → bool`

Возвращает истину, если выражение пустое

return

истина, если выражение пустое

`get_linear_part(self: arhiplexpy.QuadExpression) → arhiplexpy.LinearExpression`

Получает линейную часть выражения

return

линейная часть выражения

`get_term_coeff(self: arhiplexpy.QuadExpression, ind: int) → float`

Возвращает коэффициент по индексу элемента в выражении

param ind

индекс элемента в выражении

return

значение коэффициента

`get_term_variable1(self: arhiplexpy.QuadExpression, ind: int) → arhiplexpy.Variable`

Возвращает первую переменную по индексу элемента в выражении

param ind

индекс элемента в выражении

return

объект переменной

`get_term_variable2(self: arhiplexpy.QuadExpression, ind: int) → arhiplexpy.Variable`

Возвращает вторую переменную по индексу элемента в выражении

param ind

индекс элемента в выражении

return

объект переменной

`get_terms_count(self: arhiplexpy.QuadExpression) → int`

Возвращает количество элементов в выражении

return

количество элементов в выражении

`remove_term(self: arhiplexpy.QuadExpression, idx: int) → None`

Удаляет элемент выражения по индексу

param idx

индекс удаляемого элемента в выражении

`set_term_coeff(self: arhiplexpy.QuadExpression, ind: int, value: float) → None`

Задаёт коэффициент по индексу выражения

param ind

индекс элемента в выражении

param value

значение коэффициента в элементе

`class arhiplexpy.SolveResult`

Результат расчета

`get_best_bound(self: arhiplexpy.SolveResult) → float`

Получает значение граничной (двойственной) функции

return

значение граничной (двойственной) функции

`get_dual_value(*args, **kwargs)`

Overloaded function.

1. `get_dual_value(self: arhiplexpy.SolveResult, constraint: arhiplex::Constraint) -> float`

Получает значение dual value

param constraint

ограничение

return

значение dual value в решении

2. `get_dual_value(self: arhiplexpy.SolveResult, constraint_name: str) -> float`

Получает значение dual value по имени ограничения

param constraint_name

имя ограничения

return

значение dual value в решении

`get_expression_value(*args, **kwargs)`

Overloaded function.

1. `get_expression_value(self: arhiplexpy.SolveResult, arg0: arhiplex::QuadExpression) -> float`

Получает значение выражения

param expr

объект выражения

return

значение выражения

2. `get_expression_value(self: arhiplexpy.SolveResult, arg0: arhiplex::LinearExpression) -> float`

Получает значение выражения

param expr

объект выражения

return

значение выражения

`get_iterations_count(self: arhiplexpy.SolveResult) → int`

Получает количество итераций, проведенных в процессе расчета

return

количество итераций

`get_nodes_count(self: arhiplexpy.SolveResult) → int`

Получает количество обработанных узлов дерева решений

return

количество обработанных узлов

`get_objective_value(self: arhiplexpy.SolveResult) → float`

Получает значение целевой функции

return

значение целевой функции

`get_reduced_cost(*args, **kwargs)`

Overloaded function.

1. `get_reduced_cost(self: arhiplexpy.SolveResult, var_name: str) -> float`

Получает значение reduced cost в решении по имени

param var_name

имя переменной

return

значение reduced cost в решении

2. `get_reduced_cost(self: arhiplexpy.SolveResult, variable: arhiplex::Variable) -> float`

Получает значение reduced cost в решении

param variable

переменная

return

значение reduced cost в решении

`get_relative_gap(self: arhiplexpy.SolveResult) → float`

Получает точность решения в процентах

return

точность решения в процентах

`get_solution_status(self: arhiplexpy.SolveResult) → arhiplexpy.solution_status`

Получает статус модели в итоге расчета

return

значение статуса

`get_solve_result(self: arhiplexpy.SolveResult) → arhiplexpy.solve_result`

Получает статус процесса расчетов

return

значение статуса

`get_solve_time(self: arhiplexpy.SolveResult) → float`

Получает общее время решения

return

общее время решения [сек]

`get_variable_value(*args, **kwargs)`

Overloaded function.

1. `get_variable_value(self: arhiplexpy.SolveResult, var: arhiplexpy.Variable) -> float`

Получает значение переменной в решении

param var

объект переменной

return

значение переменной в решении

2. `get_variable_value(self: arhiplexpy.SolveResult, var_name: str) -> float`

Получает значение переменной в решении по имени

param var_name

имя переменной

return

значение переменной в решении

`write_solution(self: arhiplexpy.SolveResult, file_name: str) → None`

Записывает решение в файл

param file_name

путь к файлу решения для записи

`class arhiplexpy.Variable`

Работа с переменной - чтение/изменение имени, верхней и нижней границы, типа, а также удаление

`get_lower_bound(self: arhiplexpy.Variable) → float`

Получает нижнюю границу переменной

return

нижнюю границу переменной

`get_name(self: arhiplexpy.Variable) → str`

Получает имя переменной в модели

return

имя переменной

`get_type(self: arhiplexpy.Variable) → arhiplexpy.variable_type`

Получает тип переменной

return

тип переменной

`get_upper_bound(self: arhiplexpy.Variable) → float`

Получает верхнюю границу переменной

return

верхнюю границу переменной

`remove(self: arhiplexpy.Variable) → None`

Помечает переменную как удаленную. Переменная не будет участвовать в расчетах

`set_lower_bound(self: arhiplexpy.Variable, lower_bound: float) → None`

Задаёт нижнюю границу для переменной

param lower_bound

новая нижняя граница переменной

`set_name(self: arhiplexpy.Variable, var_name: str) → None`

Задаёт имя переменной в модели

param var_name

имя переменной

`set_type(self: arhiplexpy.Variable, type: arhiplexpy.variable_type) → None`

Задаёт новый тип переменной в модели

param type

новый тип переменной

`set_upper_bound(self: arhiplexpy.Variable, upper_bound: float) → None`

Задаёт верхнюю границу для переменной

param upper_bound

новая верхняя граница переменной

`class arhiplexpy.constraint_sense`

Members:

`equal`

`less_equal`

`greater_equal`

`property name`

`class arhiplexpy.objective_sense`

Members:

`minimize`

`maximize`

`property name`

```
class arhiplexpy.solution_status
```

Members:

invalid_solution

optimal

feasible

infeasible

unbounded

infeasible_or_unbounded

property name

```
class arhiplexpy.solve_result
```

Members:

success

fail

remote_invalid_api_key

remote_api_key_not_set

remote_time_amount_is_over

remote_time_per_calc_violated

remote_fail

property name

```
class arhiplexpy.variable_type
```

Members:

continuous

binary

integer

semicontinuous

semiinteger

property name

Параметры

Имя параметра	Описание
http_proxy	Http Proxy используемый для работы с облаком Type: string Default value:
log_to_console	Включает или отключает вывод солвера на консоль. Также применимо для удаленного расчета ArhiCloud. Type: bool Default value: true
mip_abs_gap	Абсолютная ошибка, $\text{abs}(\text{ub}-\text{lb})$, для определения степени оптимальности MIP задачи Type: floating point Default value: 0.00 Lower bound: 0.00 Upper bound: inf

continues on next page

Таблица 4.1 – продолжение с предыдущей страницы

Имя параметра	Описание
mip_feasibility_tolerance	<p>Точность достижимости МIP задачи.</p> <p>Type: floating point</p> <p>Default value: 0.00</p> <p>Lower bound: 0.00</p> <p>Upper bound: inf</p>
mip_rel_gap	<p>Относительная ошибка, $\text{abs}(\text{ub}-\text{lb})/\text{abs}(\text{ub})$, для определения степени оптимальности МIP задачи. Также применимо для удаленного расчета ArhiCloud.</p> <p>Type: floating point</p> <p>Default value: 0.00</p> <p>Lower bound: 0.00</p> <p>Upper bound: inf</p>
multiprocess_max_process_count	<p>Количество процессов для запуска</p> <p>Type: integral</p> <p>Default value: 0</p> <p>Lower bound: 0</p> <p>Upper bound: 16</p>
presolve	<p>Пресолвер: «off» или «on». Также применимо для удаленного расчета ArhiCloud.</p> <p>Type: string</p> <p>Default value: on</p>

continues on next page

Таблица 4.1 – продолжение с предыдущей страницы

Имя параметра	Описание
random_seed	<p>Начальное значение для генератора случайных чисел</p> <p>Type: integral</p> <p>Default value: 0</p> <p>Lower bound: 0</p> <p>Upper bound: 2147483647</p>
remote_timeout	<p>Максимальное время ожидания сервера удаленных вычислений (ArhiCloud), сек</p> <p>Type: integral</p> <p>Default value: 300</p> <p>Lower bound: 10</p> <p>Upper bound: 600</p>
time_limit	<p>Лимит времени в сек. Также применимо для удаленного расчета ArhiCloud.</p> <p>Type: floating point</p> <p>Default value: inf</p> <p>Lower bound: 0.00</p> <p>Upper bound: inf</p>
write_zeroes_to_solution	<p>Write zero variable values to solution file</p> <p>Type: bool</p> <p>Default value: true</p>

a

arhiplexpy, 40

A

<code>add_constant()</code> (метод <code>arhiplexy.LinearExpression</code>), 41	<code>arhiplex::Constraint::Remove</code> (C++ function), 2
<code>add_constraint()</code> (метод <code>arhiplexy.Model</code>), 43	<code>arhiplex::Constraint::SetLowerBound</code> (C++ function), 2
<code>add_constraints()</code> (метод <code>arhiplexy.Model</code>), 45	<code>arhiplex::Constraint::SetName</code> (C++ function), 3
<code>add_expression()</code> (метод <code>arhiplexy.LinearExpression</code>), 41	<code>arhiplex::Constraint::SetRange</code> (C++ function), 2
<code>add_expression()</code> (метод <code>arhiplexy.QuadExpression</code>), 55	<code>arhiplex::Constraint::SetUpperBound</code> (C++ function), 2
<code>add_mip_start_values()</code> (метод <code>arhiplexy.Model</code>), 45	<code>arhiplex::constraint_sense</code> (C++ enum), 19
<code>add_term()</code> (метод <code>arhiplexy.LinearExpression</code>), 41	<code>arhiplex::constraint_sense::equal</code> (C++ enumerator), 19
<code>add_term()</code> (метод <code>arhiplexy.QuadExpression</code>), 55	<code>arhiplex::constraint_sense::greater_equal</code> (C++ enumerator), 19
<code>add_variable()</code> (метод <code>arhiplexy.Model</code>), 46	<code>arhiplex::constraint_sense::less_equal</code> (C++ enumerator), 19
<code>arhiplex::arhiplex_exception</code> (C++ class), 1	<code>arhiplex::LinearExpression</code> (C++ class), 3
<code>arhiplex::arhiplex_exception::get_error_code</code> (C++ function), 1	<code>arhiplex::LinearExpression::AddConstant</code> (C++ function), 4
<code>arhiplex::Constraint</code> (C++ class), 1	<code>arhiplex::LinearExpression::AddExpression</code> (C++ function), 4
<code>arhiplex::Constraint::Constraint</code> (C++ function), 1	<code>arhiplex::LinearExpression::AddTerm</code> (C++ function), 4
<code>arhiplex::Constraint::GetLinearExpression</code> (C++ function), 2	<code>arhiplex::LinearExpression::CreateFreeCopy</code> (C++ function), 5
<code>arhiplex::Constraint::GetLowerBound</code> (C++ function), 2	<code>arhiplex::LinearExpression::empty</code> (C++ function), 3
<code>arhiplex::Constraint::GetName</code> (C++ function), 3	<code>arhiplex::LinearExpression::GetConstant</code> (C++ function), 4
<code>arhiplex::Constraint::GetQuadExpression</code> (C++ function), 2	<code>arhiplex::LinearExpression::GetName</code> (C++ function), 3
<code>arhiplex::Constraint::GetRange</code> (C++ function), 2	<code>arhiplex::LinearExpression::GetTermCoeff</code> (C++ function), 4
<code>arhiplex::Constraint::GetUpperBound</code> (C++ function), 2	<code>arhiplex::LinearExpression::GetTermsCount</code> (C++ function), 3
	<code>arhiplex::LinearExpression::GetTermVariable</code> (C++ function), 4

```

arhiplex::LinearExpression::LinearExpression    (C++ function), 11
    (C++ function), 3
arhiplex::LinearExpression::RemoveTerm          (C++ function), 11
    (C++ function), 5
arhiplex::LinearExpression::RemoveVariable      (C++ function), 15
    (C++ function), 5
arhiplex::LinearExpression::SetConstant        (C++ function), 7
    (C++ function), 4
arhiplex::LinearExpression::SetName            (C++ function), 7
    (C++ function), 3
arhiplex::LinearExpression::SetTermCoeff       (C++ function), 7
    (C++ function), 4
arhiplex::Model (C++ class), 7
arhiplex::Model::AddConstraint                 (C++ function), 10, 11
arhiplex::Model::AddMipStartValue              (C++ function), 14
arhiplex::Model::AddMipStartValues             (C++ function), 14
arhiplex::Model::AddVariable                   (C++ function), 9
arhiplex::Model::Clear (C++ function), 13
arhiplex::Model::ClearMipStartValues           (C++ function), 14
arhiplex::Model::GetBoolParam                  (C++ function), 8
arhiplex::Model::GetCalcUID                    (C++ function), 14
arhiplex::Model::GetConstraint                 (C++ function), 12
arhiplex::Model::GetConstraintsCount           (C++ function), 12
arhiplex::Model::GetConstrByName               (C++ function), 12
arhiplex::Model::GetDblParam                   (C++ function), 8
arhiplex::Model::GetIntParam                   (C++ function), 7
arhiplex::Model::GetName (C++ function), 14
arhiplex::Model::GetObjective                  (C++ function), 13
arhiplex::Model::GetObjectiveSense             (C++ function), 12
arhiplex::Model::GetRemoteSolveLog            (C++ function), 13
arhiplex::Model::GetStringParam               (C++ function), 8
arhiplex::Model::GetVariable                   (C++ function), 11
arhiplex::Model::GetVariableByName            (C++ function), 11
arhiplex::Model::GetVariablesCount            (C++ function), 11
arhiplex::Model::IsMip (C++ function), 15
arhiplex::Model::IsNonlinear                  (C++ function), 15
arhiplex::Model::Model (C++ function), 7
arhiplex::Model::Read (C++ function), 7
arhiplex::Model::ReadLp (C++ function), 7
arhiplex::Model::ReadMps (C++ function), 7
arhiplex::Model::Remove (C++ function), 13
arhiplex::Model::SetBoolParam                 (C++ function), 9
arhiplex::Model::SetDblParam                  (C++ function), 8
arhiplex::Model::SetIntParam                  (C++ function), 8
arhiplex::Model::SetLogCallback               (C++ function), 14
arhiplex::Model::SetLogFile                   (C++ function), 14
arhiplex::Model::SetName (C++ function), 14
arhiplex::Model::SetObjective                 (C++ function), 12
arhiplex::Model::SetObjectiveOffset           (C++ function), 12
arhiplex::Model::SetObjectiveSense            (C++ function), 12
arhiplex::Model::SetStringParam               (C++ function), 8
arhiplex::Model::Solve (C++ function), 13
arhiplex::Model::SolveRemote                  (C++ function), 13
arhiplex::Model::SolveRemoteAsync             (C++ function), 13
arhiplex::Model::Write (C++ function), 9
arhiplex::Model::WriteLp (C++ function), 9
arhiplex::Model::WriteMps                     (C++ function), 9
arhiplex::objective_sense (C++ enum), 19
arhiplex::objective_sense::maximize           (C++ enumerator), 19
arhiplex::objective_sense::minimize           (C++ enumerator), 19
arhiplex::QuadExpression (C++ class), 5
arhiplex::QuadExpression::AddExpression

```

(C++ function), 6
 arhiplex::QuadExpression::AddTerm (C++ function), 6
 arhiplex::QuadExpression::CreateFreeCopy (C++ function), 7
 arhiplex::QuadExpression::empty (C++ function), 5
 arhiplex::QuadExpression::GetTermCoeff (C++ function), 6
 arhiplex::QuadExpression::GetTermsCount (C++ function), 5
 arhiplex::QuadExpression::GetTermVariable1 (C++ function), 6
 arhiplex::QuadExpression::GetTermVariable2 (C++ function), 6
 arhiplex::QuadExpression::QuadExpression (C++ function), 5
 arhiplex::QuadExpression::RemoveTerm (C++ function), 7
 arhiplex::QuadExpression::SetTermCoeff (C++ function), 6
 arhiplex::solution_status (C++ enum), 20
 arhiplex::solution_status::feasible (C++ enumerator), 20
 arhiplex::solution_status::infeasible (C++ enumerator), 20
 arhiplex::solution_status::infeasible_or_unbounded (C++ enumerator), 20
 arhiplex::solution_status::invalid_solution (C++ enumerator), 20
 arhiplex::solution_status::optimal (C++ enumerator), 20
 arhiplex::solution_status::unbounded (C++ enumerator), 20
 arhiplex::solve_result (C++ enum), 19
 arhiplex::solve_result::fail (C++ enumerator), 19
 arhiplex::solve_result::remote_api_key_not_set (C++ enumerator), 19
 arhiplex::solve_result::remote_fail (C++ enumerator), 20
 arhiplex::solve_result::remote_invalid_api_key (C++ enumerator), 19
 arhiplex::solve_result::remote_time_amount_is_overflow (C++ enumerator), 20
 arhiplex::solve_result::remote_time_per_candidate_exceeded (C++ enumerator), 20
 arhiplex::solve_result::success (C++ enumerator), 19
 arhiplex::SolveResult (C++ class), 15
 arhiplex::SolveResult::GetBestBoundValue (C++ function), 16
 arhiplex::SolveResult::GetDualValue (C++ function), 16
 arhiplex::SolveResult::GetExpressionValue (C++ function), 17
 arhiplex::SolveResult::GetIterationsCount (C++ function), 15
 arhiplex::SolveResult::GetObjectiveFunctionValue (C++ function), 15
 arhiplex::SolveResult::GetProcessedNodesCount (C++ function), 15
 arhiplex::SolveResult::GetReducedCost (C++ function), 16, 17
 arhiplex::SolveResult::GetRelativeGap (C++ function), 15
 arhiplex::SolveResult::GetSolutionStatus (C++ function), 15
 arhiplex::SolveResult::GetSolveResult (C++ function), 15
 arhiplex::SolveResult::GetSolveTime (C++ function), 16
 arhiplex::SolveResult::GetVariableValue (C++ function), 16
 arhiplex::SolveResult::WriteSolution (C++ function), 17
 arhiplex::Variable (C++ class), 17
 arhiplex::Variable::GetLowerBound (C++ function), 18
 arhiplex::Variable::GetName (C++ function), 17
 arhiplex::Variable::GetType (C++ function), 17
 arhiplex::Variable::GetUpperBound (C++ function), 18
 arhiplex::Variable::Remove (C++ function), 18
 arhiplex::Variable::SetLowerBound (C++ function), 18
 arhiplex::Variable::SetName (C++ function), 17
 arhiplex::Variable::SetType (C++ function), 18
 arhiplex::Variable::SetUpperBound (C++ function), 18
 arhiplex::variable_type (C++ enum), 18
 arhiplex::variable_type::binary (C++ enumerator), 18
 arhiplex::variable_type::continuous (C++ enumerator), 18
 arhiplex::variable_type::integer (C++

- enumerator*), 18
- `arhiplex::variable_type::semicontinuous` (*C++ enumerator*), 18
- `arhiplex::variable_type::semiinteger` (*C++ enumerator*), 19
- `ArhiplexException`, 40
- `arhiplexpy`
module, 40
- ## B
- `binary_var_list()` (метод *arhiplexpy.Model*), 46
- ## C
- `clear()` (метод *arhiplexpy.Model*), 46
- `clear_mip_start_values()` (метод *arhiplexpy.Model*), 46
- `Constraint` (класс в *arhiplexpy*), 40
- `constraint_sense` (класс в *arhiplexpy*), 60
- `continuous_var_list()` (метод *arhiplexpy.Model*), 46
- `copy()` (метод *arhiplexpy.LinearExpression*), 42
- `copy()` (метод *arhiplexpy.QuadExpression*), 55
- ## E
- `empty()` (метод *arhiplexpy.LinearExpression*), 42
- `empty()` (метод *arhiplexpy.QuadExpression*), 56
- ## G
- `get_best_bound()` (метод *arhiplexpy.SolveResult*), 57
- `get_bool_param()` (метод *arhiplexpy.Model*), 48
- `get_calc_uid()` (метод *arhiplexpy.Model*), 48
- `get_constant()` (метод *arhiplexpy.LinearExpression*), 42
- `get_constraint()` (метод *arhiplexpy.Model*), 48
- `get_constraints_count()` (метод *arhiplexpy.Model*), 48
- `get_dbl_param()` (метод *arhiplexpy.Model*), 49
- `get_dual_value()` (метод *arhiplexpy.SolveResult*), 57
- `get_expression_value()` (метод *arhiplexpy.SolveResult*), 57
- `get_int_param()` (метод *arhiplexpy.Model*), 49
- `get_iterations_count()` (метод *arhiplexpy.SolveResult*), 58
- `get_linear_expression()` (метод *arhiplexpy.Constraint*), 40
- `get_linear_part()` (метод *arhiplexpy.QuadExpression*), 56
- `get_lower_bound()` (метод *arhiplexpy.Constraint*), 40
- `get_lower_bound()` (метод *arhiplexpy.Variable*), 59
- `get_name()` (метод *arhiplexpy.Constraint*), 40
- `get_name()` (метод *arhiplexpy.LinearExpression*), 42
- `get_name()` (метод *arhiplexpy.Model*), 49
- `get_name()` (метод *arhiplexpy.Variable*), 59
- `get_nodes_count()` (метод *arhiplexpy.SolveResult*), 58
- `get_objective()` (метод *arhiplexpy.Model*), 49
- `get_objective_sense()` (метод *arhiplexpy.Model*), 49
- `get_objective_value()` (метод *arhiplexpy.SolveResult*), 58
- `get_quad_expression()` (метод *arhiplexpy.Constraint*), 40
- `get_quad_part()` (метод *arhiplexpy.LinearExpression*), 42
- `get_range()` (метод *arhiplexpy.Constraint*), 40
- `get_reduced_cost()` (метод *arhiplexpy.SolveResult*), 58
- `get_relative_gap()` (метод *arhiplexpy.SolveResult*), 58
- `get_remote_solve_log()` (метод *arhiplexpy.Model*), 49
- `get_solution_status()` (метод *arhiplexpy.SolveResult*), 58
- `get_solve_result()` (метод *arhiplexpy.SolveResult*), 58
- `get_solve_time()` (метод *arhiplexpy.SolveResult*), 59
- `get_string_param()` (метод *arhiplexpy.Model*), 49
- `get_term_coeff()` (метод *arhiplexpy.LinearExpression*), 42
- `get_term_coeff()` (метод *arhiplexpy.QuadExpression*), 56
- `get_term_variable()` (метод *arhiplexpy.LinearExpression*), 42

<code>get_term_variable1()</code> (<i>метод</i> <code>arhiplexpy.QuadExpression</code>), 56	<code>PhonyNameDueToError::AddExpression</code> (<i>C++ function</i>), 24, 26
<code>get_term_variable2()</code> (<i>метод</i> <code>arhiplexpy.QuadExpression</code>), 56	<code>PhonyNameDueToError::AddMipStartValue</code> (<i>C++ function</i>), 33
<code>get_terms_count()</code> (<i>метод</i> <code>arhiplexpy.LinearExpression</code>), 42	<code>PhonyNameDueToError::AddMipStartValues</code> (<i>C++ function</i>), 33
<code>get_terms_count()</code> (<i>метод</i> <code>arhiplexpy.QuadExpression</code>), 56	<code>PhonyNameDueToError::AddTerm</code> (<i>C++ function</i>), 24, 25
<code>get_type()</code> (<i>метод</i> <code>arhiplexpy.Variable</code>), 59	<code>PhonyNameDueToError::AddVariable</code> (<i>C++ function</i>), 29
<code>get_upper_bound()</code> (<i>метод</i> <code>arhiplexpy.Constraint</code>), 40	<code>PhonyNameDueToError::binary</code> (<i>C++ enumerator</i>), 37
<code>get_upper_bound()</code> (<i>метод</i> <code>arhiplexpy.Variable</code>), 60	<code>PhonyNameDueToError::Clear</code> (<i>C++ function</i>), 33
<code>get_variable()</code> (<i>метод</i> <code>arhiplexpy.Model</code>), 49	<code>PhonyNameDueToError::ClearMipStartValues</code> (<i>C++ function</i>), 33
<code>get_variable_value()</code> (<i>метод</i> <code>arhiplexpy.SolveResult</code>), 59	<code>PhonyNameDueToError::ClearObjective</code> (<i>C++ function</i>), 32
<code>get_variables_count()</code> (<i>метод</i> <code>arhiplexpy.Model</code>), 50	<code>PhonyNameDueToError::continuous</code> (<i>C++ enumerator</i>), 37
	<code>PhonyNameDueToError::CreateFreeCopy</code> (<i>C++ function</i>), 24, 26
<code>integer_var_list()</code> (<i>метод</i> <code>arhiplexpy.Model</code>), 50	<code>PhonyNameDueToError::equal</code> (<i>C++ enumerator</i>), 38
<code>is_mip()</code> (<i>метод</i> <code>arhiplexpy.Model</code>), 51	<code>PhonyNameDueToError::fail</code> (<i>C++ enumerator</i>), 38
<code>is_nonlinear()</code> (<i>метод</i> <code>arhiplexpy.Model</code>), 51	<code>PhonyNameDueToError::feasible</code> (<i>C++ enumerator</i>), 39
L	<code>PhonyNameDueToError::GetBestBoundValue</code> (<i>C++ function</i>), 35
<code>LinearExpression</code> (<i>класс в arhiplexpy</i>), 41	<code>PhonyNameDueToError::GetBoolParam</code> (<i>C++ function</i>), 28
M	<code>PhonyNameDueToError::GetConstant</code> (<i>C++ function</i>), 23
<code>Model</code> (<i>класс в arhiplexpy</i>), 43	<code>PhonyNameDueToError::GetConstraint</code> (<i>C++ function</i>), 31
<code>module</code> <code>arhiplexpy</code> , 40	<code>PhonyNameDueToError::GetConstraintsCount</code> (<i>C++ function</i>), 31
N	<code>PhonyNameDueToError::GetDblParam</code> (<i>C++ function</i>), 27
<code>name</code> (<i>arhiplexpy.constraint_sense property</i>), 60	<code>PhonyNameDueToError::GetDualValue</code> (<i>C++ function</i>), 35
<code>name</code> (<i>arhiplexpy.objective_sense property</i>), 60	<code>PhonyNameDueToError::GetExpressionValue</code> (<i>C++ function</i>), 36
<code>name</code> (<i>arhiplexpy.solution_status property</i>), 61	<code>PhonyNameDueToError::GetIntParam</code> (<i>C++ function</i>), 27
<code>name</code> (<i>arhiplexpy.solve_result property</i>), 61	<code>PhonyNameDueToError::GetIterationsCount</code> (<i>C++ function</i>), 34
<code>name</code> (<i>arhiplexpy.variable_type property</i>), 61	<code>PhonyNameDueToError::GetLinearExpression</code> (<i>C++ function</i>), 21
O	<code>PhonyNameDueToError::GetLinearPart</code>
<code>objective_sense</code> (<i>класс в arhiplexpy</i>), 60	
P	
<code>PhonyNameDueToError::AddConstant</code> (<i>C++ function</i>), 24	
<code>PhonyNameDueToError::AddConstraint</code> (<i>C++ function</i>), 29, 30	

<i>(C++ function)</i> , 26	PhonyNameDueToError::greater_equal <i>(C++ enumerator)</i> , 38
PhonyNameDueToError::GetLowerBound <i>(C++ function)</i> , 22, 37	PhonyNameDueToError::infeasible <i>(C++ enumerator)</i> , 39
PhonyNameDueToError::GetObjective <i>(C++ function)</i> , 32	PhonyNameDueToError::infeasible_or_unbounded
PhonyNameDueToError::GetObjectiveFunctionValue <i>(C++ enumerator)</i> , 39 <i>(C++ function)</i> , 35	PhonyNameDueToError::integer <i>(C++ enumerator)</i> , 37
PhonyNameDueToError::GetObjectiveSense <i>(C++ function)</i> , 31	PhonyNameDueToError::invalid_solution
PhonyNameDueToError::GetProcessedNodesCount <i>(C++ enumerator)</i> , 39 <i>(C++ function)</i> , 34	PhonyNameDueToError::IsMip <i>(C++ function)</i> , 34
PhonyNameDueToError::GetQuadExpression <i>(C++ function)</i> , 21	PhonyNameDueToError::IsNonlinear <i>(C++ function)</i> , 34
PhonyNameDueToError::GetQuadObjective <i>(C++ function)</i> , 32	PhonyNameDueToError::less_equal <i>(C++ enumerator)</i> , 38
PhonyNameDueToError::GetQuadPart <i>(C++ function)</i> , 24	PhonyNameDueToError::LinearExpression <i>(C++ function)</i> , 23
PhonyNameDueToError::GetRange <i>(C++ function)</i> , 22	PhonyNameDueToError::maximize <i>(C++ enumerator)</i> , 38
PhonyNameDueToError::GetReducedCost <i>(C++ function)</i> , 35, 36	PhonyNameDueToError::minimize <i>(C++ enumerator)</i> , 38
PhonyNameDueToError::GetRelativeGap <i>(C++ function)</i> , 34	PhonyNameDueToError::Model <i>(C++ function)</i> , 26
PhonyNameDueToError::GetSolutionStatus <i>(C++ function)</i> , 34	PhonyNameDueToError::optimal <i>(C++ enumerator)</i> , 39
PhonyNameDueToError::GetSolveResult <i>(C++ function)</i> , 34	PhonyNameDueToError::QuadExpression <i>(C++ function)</i> , 25
PhonyNameDueToError::GetSolveTime <i>(C++ function)</i> , 35	PhonyNameDueToError::Read <i>(C++ function)</i> , 26
PhonyNameDueToError::GetStringParam <i>(C++ function)</i> , 28	PhonyNameDueToError::ReadLp <i>(C++ function)</i> , 27
PhonyNameDueToError::GetTermCoeff <i>(C++ function)</i> , 23, 25	PhonyNameDueToError::ReadMps <i>(C++ function)</i> , 26
PhonyNameDueToError::GetTermsCount <i>(C++ function)</i> , 23, 25	PhonyNameDueToError::remote_api_key_not_set <i>(C++ enumerator)</i> , 38
PhonyNameDueToError::GetTermVariable <i>(C++ function)</i> , 23	PhonyNameDueToError::remote_fail <i>(C++ enumerator)</i> , 39
PhonyNameDueToError::GetTermVariable1 <i>(C++ function)</i> , 25	PhonyNameDueToError::remote_invalid_api_key <i>(C++ enumerator)</i> , 38
PhonyNameDueToError::GetTermVariable2 <i>(C++ function)</i> , 25	PhonyNameDueToError::remote_time_amount_is_over <i>(C++ enumerator)</i> , 39
PhonyNameDueToError::GetUpperBound <i>(C++ function)</i> , 22, 36	PhonyNameDueToError::remote_time_per_calc_violated <i>(C++ enumerator)</i> , 39
PhonyNameDueToError::GetVariable <i>(C++ function)</i> , 30, 31	PhonyNameDueToError::Remove <i>(C++ function)</i> , 21, 33, 37
PhonyNameDueToError::GetVariablesCount <i>(C++ function)</i> , 30	PhonyNameDueToError::RemoveTerm <i>(C++ function)</i> , 24, 26
PhonyNameDueToError::GetVariableValue <i>(C++ function)</i> , 35	PhonyNameDueToError::RemoveVariable <i>(C++ function)</i> , 24
PhonyNameDueToError::GetVarType <i>(C++ function)</i> , 37	PhonyNameDueToError::semicontinuous

(C++ enumerator), 37

PhonyNameDueToError::semiinteger (C++ enumerator), 38

PhonyNameDueToError::SetBoolParam (C++ function), 28

PhonyNameDueToError::SetConstant (C++ function), 23

PhonyNameDueToError::SetDblParam (C++ function), 28

PhonyNameDueToError::SetIntParam (C++ function), 28

PhonyNameDueToError::SetLogFile (C++ function), 33

PhonyNameDueToError::SetLowerBound (C++ function), 22, 36

PhonyNameDueToError::SetName (C++ function), 21, 29, 37

PhonyNameDueToError::SetObjective (C++ function), 31, 32

PhonyNameDueToError::SetObjectiveOffset (C++ function), 31

PhonyNameDueToError::SetObjectiveSense (C++ function), 31

PhonyNameDueToError::SetRange (C++ function), 22

PhonyNameDueToError::SetStringParam (C++ function), 28

PhonyNameDueToError::SetTermCoeff (C++ function), 23, 25

PhonyNameDueToError::SetType (C++ function), 37

PhonyNameDueToError::SetUpperBound (C++ function), 22, 36

PhonyNameDueToError::Solve (C++ function), 32

PhonyNameDueToError::SolveRemote (C++ function), 32

PhonyNameDueToError::SolveRemoteAsync (C++ function), 32

PhonyNameDueToError::success (C++ enumerator), 38

PhonyNameDueToError::unbounded (C++ enumerator), 39

PhonyNameDueToError::Write (C++ function), 27

PhonyNameDueToError::WriteLp (C++ function), 27

PhonyNameDueToError::WriteMps (C++ function), 27

PhonyNameDueToError::WriteSolution (C++ function), 36

Q

QuadExpression (класс в arhiplexy), 55

R

read() (метод arhiplexy.Model), 51

read_lp() (метод arhiplexy.Model), 52

read_mps() (метод arhiplexy.Model), 52

remove() (метод arhiplexy.Constraint), 41

remove() (метод arhiplexy.Model), 52

remove() (метод arhiplexy.Variable), 60

remove_term() (метод arhiplexy.LinearExpression), 43

remove_term() (метод arhiplexy.QuadExpression), 56

remove_variable() (метод arhiplexy.LinearExpression), 43

S

set_bool_param() (метод arhiplexy.Model), 52

set_constant() (метод arhiplexy.LinearExpression), 43

set_dbl_param() (метод arhiplexy.Model), 52

set_int_param() (метод arhiplexy.Model), 52

set_log_callback() (метод arhiplexy.Model), 53

set_log_file() (метод arhiplexy.Model), 53

set_lower_bound() (метод arhiplexy.Constraint), 41

set_lower_bound() (метод arhiplexy.Variable), 60

set_name() (метод arhiplexy.Constraint), 41

set_name() (метод arhiplexy.LinearExpression), 43

set_name() (метод arhiplexy.Model), 53

set_name() (метод arhiplexy.Variable), 60

set_objective() (метод arhiplexy.Model), 53

set_objective_offset() (метод arhiplexy.Model), 53

set_objective_sense() (метод arhiplexy.Model), 53

set_range() (метод arhiplexy.Constraint), 41

set_string_param() (метод arhiplexy.Model), 53

set_term_coeff() (метод arhiplexy.LinearExpression), 43

`set_term_coeff()` (метод *arhiplexpy.QuadExpression*), 56
`set_type()` (метод *arhiplexpy.Variable*), 60
`set_upper_bound()` (метод *arhiplexpy.Constraint*), 41
`set_upper_bound()` (метод *arhiplexpy.Variable*), 60
`solution_status` (класс в *arhiplexpy*), 60
`solve()` (метод *arhiplexpy.Model*), 54
`solve_remote()` (метод *arhiplexpy.Model*), 54
`solve_remote_async()` (метод *arhiplexpy.Model*), 54
`solve_result` (класс в *arhiplexpy*), 61
`SolveResult` (класс в *arhiplexpy*), 57
`sum()` (метод *arhiplexpy.Model*), 54

V

`Variable` (класс в *arhiplexpy*), 59
`variable_type` (класс в *arhiplexpy*), 61

W

`write()` (метод *arhiplexpy.Model*), 54
`write_lp()` (метод *arhiplexpy.Model*), 54
`write_mps()` (метод *arhiplexpy.Model*), 54
`write_solution()` (метод *arhiplexpy.SolveResult*), 59