object-oriented-programming-2

Создано системой Doxygen 1.9.4

Глава 1

Алфавитный указатель классов

1.1 Классы

Классы с их кратким описанием.

Глава 2

Список файлов

2.1 Файлы

Полный список документированных файлов.

Глава 3

Классы

3.1 Класс Bike

Класс для р.2.

Открытые члены

- Bike (std::string brand="unknown", std::string bikeType="unknown", std::string breaks↩ Type="unknown", int wheelCount=2, float wheelDiameter=26, bool isAbsorber=0, bool is↩ Adult=1)
- Bike (const Bike &other)=default
- Bike & operator= (const Bike &other)=default
- ~Bike ()=default
- std::string GetBrand () const noexcept
- int GetWheelCount () const noexcept
- float GetWheelDiameter () const noexcept
- std::string GetBikeType () const noexcept
- std::string GetBreaksType () const noexcept
- bool GetIsAbsorber () const noexcept
- bool GetIsAdult () const noexcept
- bool operator== (const Bike &other) const noexcept
- bool operator> (const Bike &other) const noexcept
- bool operator< (const Bike &other) const noexcept
- bool operator>= (const Bike &other) const noexcept
- bool operator<= (const Bike &other) const noexcept

Закрытые данные

- std::string _brand
- std::string _bikeType
- std::string _breaksType
- int _wheelCount
- float _wheelDiameter
- bool _isAbsorber
- bool _isAdult

Друзья

- std::ostream & operator<< (std::ostream &output, const Bike &bike)

3.1.1 Подробное описание

Класс для p.2.

Класс был создан по таблице 1.2 из методического пособия по ООП/2.

3.1.2 Конструктор(ы)

3.1.2.1 Bike() [1/2]

```
Bike::Bike (
            std::string brand = "unknown",
            std::string bikeType = "unknown",
            std::string breaksType = "unknown",
            int wheelCount = 2,
            float wheelDiameter = 26,
            bool isAbsorber = 0,
            bool isAdult = 1 )   [inline]
```

Конструктор по умолчанию.

Аргументы

brand	
bikeType	
breaksType	
wheelCount	
wheelDiameter	
isAbsorber	
isAdult	

```
42                          {
43          _brand = brand;
44          _bikeType = bikeType;
45          _breaksType = breaksType;
46          _wheelCount = wheelCount;
47          _wheelDiameter = wheelDiameter;
48          _isAbsorber = isAbsorber;
49          _isAdult = isAdult;
50      }
```

3.1.2.2 Bike() [2/2]

```
Bike::Bike (
            const Bike & other )   [default]
```

Конструктор копий.

Предупреждения

Опеределен компилятором.

Аргументы

other	

3.1.2.3 ~Bike()

Bike::~Bike ( ) [default]

Деструктор.

Предупреждения

Деструктор определен компилятором и не является виртуальным.

3.1.3 Методы

3.1.3.1 GetBikeType()

std::string Bike::GetBikeType ( ) const [inline], [noexcept]
90 { return _bikeType; }

3.1.3.2 GetBrand()

std::string Bike::GetBrand ( ) const [inline], [noexcept]
84 { return _brand; }

3.1.3.3 GetBreaksType()

std::string Bike::GetBreaksType ( ) const [inline], [noexcept]
92 { return _breaksType; }

3.1.3.4 GetIsAbsorber()

```
bool Bike::GetIsAbsorber ( ) const    [inline], [noexcept]
94 { return _isAbsorber; }
```

3.1.3.5 GetIsAdult()

```
bool Bike::GetIsAdult ( ) const    [inline], [noexcept]
96 { return _isAdult; }
```

3.1.3.6 GetWheelCount()

```
int Bike::GetWheelCount ( ) const    [inline], [noexcept]
86 { return _wheelCount; }
```

3.1.3.7 GetWheelDiameter()

```
float Bike::GetWheelDiameter ( ) const    [inline], [noexcept]
88 { return _wheelDiameter; }
```

3.1.3.8 operator<()

```
bool Bike::operator< (
                const Bike & other ) const    [inline], [noexcept]
```

Перегрузка оператора <.

Аргументы

other	

Возвращает

Противоположное значение оператора>

```
177                                         {
178         return !(*this > other);
179     }
```

3.1.3.9 operator<=()

```
bool Bike::operator<= (
                const Bike & other ) const   [inline], [noexcept]
185                                          {
186     return ( !(*this > other) || *this == other );
187 }
```

3.1.3.10 operator=()

```
Bike & Bike::operator= (
                const Bike & other )   [default]
```

Перегрузка оператора =.

Предупреждения

Определен компилятором.

Аргументы

other	

Возвращает

Новый объект класса, поля которого равны полям входящего объекта.

3.1.3.11 operator==()

```
bool Bike::operator== (
                const Bike & other ) const   [inline], [noexcept]
```

Перегрузка оператора ==.

Аргументы

other	

Возвращает

Возвращает true, если поля экземплятров класса совпадают, в противном случае - false.

```
107                                          {
108     if ( _brand == other._brand &&
109         _bikeType == other._bikeType &&
110         _breaksType == other._breaksType &&
111         _wheelCount == other._wheelCount &&
112         _wheelDiameter == other._wheelDiameter &&
```

```
113             _isAbsorber == other._isAbsorber &&
114             _isAdult == other._isAdult ) {
115             return true;
116         }
117 
118 
119         return false;
120     }
```

3.1.3.12 operator>()

bool Bike::operator> (
const Bike & other) const [inline], [noexcept]

Перегрузка оператора >.

Сравнивает объекты класса Bike в соответствии с приоритетом полей класса.

Цепочка приоритета: _wheelDiameter -> _wheelCount -> _brand(lexi-graph) -> ...

Аргументы

other	

Возвращает

true, если поле объекта *this имеет больший приоритет, если приоритеты равны, сравнение спускается по цепочки приоритетов, иначе false.

```
133                                             {
134         if ( _wheelDiameter > other._wheelDiameter ) {
135             return true;
136         }
137         else if ( _wheelDiameter == other._wheelDiameter ) {
138             if ( _wheelCount > other._wheelCount ) {
139                 return true;
140             }
141             else if ( _wheelCount == other._wheelCount ) {
142 
143                 std::string brandLowerCase = _brand;
144                 std::transform(brandLowerCase.begin(), brandLowerCase.end(), brandLowerCase.begin(),
145                                [](unsigned char c) { return std::tolower(c); }); //< преобразование поля *this._brand в нижний
    регистр
146 
147                 std::string otherBrandLowerCasae = other._brand;
148                 std::transform(otherBrandLowerCasae.begin(), otherBrandLowerCasae.end(), otherBrandLowerCasae.begin(),
149                                [](unsigned char c) { return std::tolower(c); }); //< преобразование поля other._brand в нижний
    регистр
150 
151                 // other way
152                 /*std::string brandLowerCase = _brand;
153 std::for_each(brandLowerCase.begin(), brandLowerCase.end(),
154 [](unsigned char c) { return std::tolower(c); });
155 
156 std::string otherBrandLowerCasae = other._brand;
157 std::for_each(otherBrandLowerCasae.begin(), otherBrandLowerCasae.end(),
158 [](unsigned char c) { return std::tolower(c); });*/
159 
160                 if ( brandLowerCase > otherBrandLowerCasae ) {
161                     return true;
162                 }
163             }
164         }
165 
166 
167         return false;
168     }
```

3.1.3.13 operator>=()

```
bool Bike::operator>= (
                const Bike & other ) const   [inline], [noexcept]
181                                          {
182     return (*this > other || *this == other);
183 }
```

3.1.4 Документация по друзьям класса и функциям, относящимся к классу

3.1.4.1 operator<<

```
std::ostream & operator<< (
                std::ostream & output,
                const Bike & bike )   [friend]
```

Перегрузка оператора<<.

Аргументы

output	
bike	

Возвращает

output

```
220                                                 {
221     if ( typeid(output).name() != typeid(std::ofstream).name() ) {
222         output « "{ " « bike.GetWheelDiameter() « "; " « bike.GetWheelCount() « "; "
223             « bike.GetBrand() « "; " « bike.GetBikeType() « "; " « bike.GetBreaksType() « "; "
224             « bike.GetIsAbsorber() « "; " « bike.GetIsAdult() « " }";
225     }
226     else {
227         output « nline « bike.GetWheelDiameter() « " " « bike.GetWheelCount() « " "
228             « bike.GetBrand() « " " « bike.GetBikeType() « " " « bike.GetBreaksType() « " "
229             « bike.GetIsAbsorber() « " " « bike.GetIsAdult() « nline;
230     }
231
232
233     return output;
234 }
```

Объявления и описания членов класса находятся в файле:

- C:/Users/averu/Documents/git_local/programming-practice/object-oriented-programming-2/practice1[list].cpp

Глава 4

Файлы

4.1 Файл practice1[list].cpp

Практическая работа 1.

```
#include <iostream>
#include <fstream>
#include <list>
#include <iterator>
#include <algorithm>
```

Классы

- class Bike

  Класс для р.2.

Макросы

- #define nline '\n'

Функции

- std::ostream & operator<< (std::ostream &output, const Bike &bike)
- template<class Type >
  void push_ (std::list< Type > &list, const Type &object)
- void push_ (std::list< Bike > &bikeList, const Bike &bike)
- template<class Type >
  Type pop_ (std::list< Type > &list, const int &pos=0)
- template<class Type >
  std::list< Type > filter (const std::list< Type > &list, int(*func_key)(const Type &))
- template<class Type >
  void print (const std::list< Type > &list)
- int IsFactorialOfEven (const int &value_)
- int main ()

4.1.1 Подробное описание

Практическая работа 1.

Автор

Sirazetdinov Rustem

Дата

August 2022

4.1.2 Функции

4.1.2.1 filter()

```
template<class Type >
std::list< Type > filter (
                const std::list< Type > & list,
                int(*)(const Type &) func_key )
```

Фильтрация объектов контейнера по функции-ключу.

Аргументы

list	
*func_key	

Возвращает

Новый контейнер, элементы которого удовлетворяют функции-ключу.

```
340 {
341     std::list<Type> ListResult;
342 
343     // for (std::list<Type>::iterator it_pos = list.begin(); it_pos != list.end(); it_pos++) {...}
344     for ( auto object : list ) {
345         if ( func_key(object) ) { ListResult.push_back(object); }
346     }
347 
348 
349     return ListResult;
350 }
```

4.1.2.2 IsFactorialOfEven()

```
int IsFactorialOfEven (
                const int & value_ )
```

Определение факторила четного числа.

Функция опередеяет является ли заданное число факториалом четного число, и находит это число, если это так.

Аргументы

value↩ _	

Возвращает

0, если входное число не является факториалом четного числа, иначе само число, факториалом которого является входное число.

```
378                                     {
379     int value = value_; int div = 1;
380
381     while ( value > 1 ) {
382         div++;
383
384         if ( value % div == 0 ) {
385             value = value / div;
386         }
387         else { break; }
388     }
389
390
391     if ( value == 1 && div % 2 == 0 ) { return div; }
392     return 0;
393 }
```

4.1.2.3 main()

```
int main ( )
398 {
399     // int value = 2;
400     // while ( value != -1) {
401     //  std::cin » value;
402     //  std::cout « IsFactorialOfEven(value) « nline;
403     // }
404
405
406
407     // -------------- p.1 --------------
408     // Создаем лист с элементами типа ‘int‘
409     std::list<int> myList = {215, 507, 668, 680, 1004, 1207, 1550, 2854,
410         2972, 3091, 3209, 3706, 4078, 4482, 4925, 5458, 5892, 6476, 6896,
411         7076, 7268, 7373, 8168, 8443, 9406};
412
413     // Проверка функции ‘print‘ для объектов типа std::list<Type>
414     print(myList);
415
416
417     // Добавим в лист некоторое количество элементов. Проверим, останется ли
418     // он отсортированным
419     push_(myList, 2);            // = 2!
420     push_(myList, 24);           // = 4!
421     push_(myList, 720);          // = 6!
422     push_(myList, 40320);        // = 8!
423     push_(myList, 3628800);      // = 10!
424     push_(myList, 479001600);    // = 12!
425     push_(myList, 362880);       // = 9!
426     push_(myList, 120);          // = 5!
427     push_(myList, 1);            // = 1!
428
429     if ( std::is_sorted(myList.begin(), myList.end()) ) { std::cout « "SORTED" « nline; }
430     else { std::cout « "UNSORTED" « nline; }
431
432     print(myList);
433
434     // Удалим из листа несколько значений
435     auto hValue_temp = pop_(myList, 1); std::cout « nline « hValue_temp « " ";
436     hValue_temp = pop_(myList, 9); std::cout « nline « hValue_temp « " ";
437     hValue_temp = pop_(myList, myList.size() - 1); std::cout « nline « hValue_temp « " ";
438     std::cout « nline « nline;
439
440     if ( std::is_sorted(myList.begin(), myList.end()) ) { std::cout « "SORTED" « nline; }
```

```
441     else { std::cout « "UNSORTED" « nline; }
442
443     print(myList);
444
445     // Выделим из нашего листа новый, элементы которого представляют собой
446     // факториалы четных чисел.
447     // int (*func_key)(const int &) = nullptr; func_key = &IsFactorialOfEven;
448     int (*func_key)(const int &) = IsFactorialOfEven;
449
450     std::list<int> newList = filter(myList, func_key);
451     print(newList);
452
453     myList.clear();
454
455
456     // -------------- p.2 --------------
457     std::list<Bike> bikeList;
458
459     Bike bikeSimple("simple", "mountain", "mechamnical", 2, 24, false, false);
460     Bike bikeSimpleUPG("Simple", "mountain", "disc", 2, 24, false, false);
461     // std::cout « nline « bikeSimple « nline « bikeSimpleUPG « nline;
462     // bikeSimple > bikeSimpleUPG ? std::cout « nline « "1" : std::cout « nline « "2";
463
464     Bike bikeUltra("ULTRA", "road", "hydDisc", 2, 29, true, true);
465     Bike bikeMedium("Medium", "urban", "disc", 2, 27, true, true);
466     Bike bikeHigh("High", "mountain", "disc", 2, 27.5, true, true);
467
468     push_(bikeList, bikeSimple);
469     push_(bikeList, bikeSimpleUPG);
470     push_(bikeList, bikeUltra);
471     push_(bikeList, bikeHigh);
472     push_(bikeList, bikeMedium);
473
474     print(bikeList);
475
476     std::cout « pop_(bikeList, 2);
477
478
479     return 0;
480 }
```

4.1.2.4 operator<<()

```
std::ostream & operator<< (
                std::ostream & output,
                const Bike & bike )
```

Перегрузка оператора<<.

Аргументы

output	
bike	

Возвращает

output

```
220                                             {
221     if ( typeid(output).name() != typeid(std::ofstream).name() ) {
222         output « "{ " « bike.GetWheelDiameter() « "; " « bike.GetWheelCount() « "; "
223             « bike.GetBrand() « "; " « bike.GetBikeType() « "; " « bike.GetBreaksType() « "; "
224             « bike.GetIsAbsorber() « "; " « bike.GetIsAdult() « " }";
225     }
226     else {
227         output « nline « bike.GetWheelDiameter() « " " « bike.GetWheelCount() « " "
228             « bike.GetBrand() « " " « bike.GetBikeType() « " " « bike.GetBreaksType() « " "
229             « bike.GetIsAbsorber() « " " « bike.GetIsAdult() « nline;
230     }
231
```

```
232
233     return output;
234 }
```

4.1.2.5 pop_()

```
template<class Type >
Type pop_ (
                std::list< Type > & list,
                const int & pos = 0 )
```

Удаление элемента из контейнера std::list.

Функция удаляет элемент из контейнера и возвращает элемент, который имеет наибольший приоритет, поскольку контейнер отсортирован, наибольший приоритет имеет объект, стоящий самым первым. Если после удаления элемента из контейнера, контейнер оказывается пустым, то возвращается последний удаленный объект.

Аргументы

list	
pos	

Возвращает

Объект с наибольшим приоритетом.

Предупреждения

Объект типа Type обязательно должен иметь конструктор по умолчанию, иначе UB!

```
300                                                 {
301
302     // инициализация по умолчанию +-
303     Type object_r_temp;
304
305     if ( pos == 0 ) {
306         object_r_temp = list.front();
307         list.pop_front();
308     }
309     else if ( pos == list.size() ) {
310         object_r_temp = list.back();
311         list.pop_back();
312     }
313     else {
314         auto it_pos = std::next(list.begin(), pos);
315         // or / auto it_pos = list.begin(); std::advance(it_pos, pos);
316         // or / std::list<Type>::iterator it_pos = std::next(list.begin(), pos);
317         object_r_temp = *it_pos;
318         list.erase(it_pos);
319     }
320
321
322     if ( !list.empty() ) {
323         return list.back(); // != return *list.end() == nullptr;
324     }
325     else {
326         return object_r_temp;
327     }
328 }
```

4.1.2.6 print()

```
template<class Type >
void print (
                const std::list< Type > & list )
```

Вывод содержимого контейнера.

Отправляет каждый элемент контейнера в поток вывода std::ostream.

Аргументы

list	

```
359                                                 {
360     std::cout « "{";
361     // for (std::list<Type>::iterator it_pos = list.begin(); it_pos != list.end(); it_pos++) {...}
362     for ( auto object : list ) {
363         std::cout « " " « object;
364     }
365     std::cout « "}; " « nline;
366 }
```

4.1.2.7 push_() [1/2]

```
void push_ (
                std::list< Bike > & bikeList,
                const Bike & bike )
```

\ Спецификация шаблона функции push_.

Данная функция является спецификацией шаблона функции. Она предназначена для работы с экземплярами класса Bike, поскольку требуется расположить экземпляры этого класса в контейнере с сортировкой по убыванию.

Аргументы

bikeList	
bike	

```
271                                                 {
272     auto it_pos = bikeList.begin();
273     // std::list<Type>::iterator it_pos = list.begin();
274
275     while ( it_pos != bikeList.end() ) {
276         // >= лучше чем >, т.к. при > вставка сильно замедляется,
277         // если все объекты в списке равны между собой
278         if ( *it_pos <= bike ) { break; }
279         it_pos++;
280     }
281
282     bikeList.insert(it_pos, bike);
283 }
```

4.1.2.8 push_() [2/2]

```
template<class Type >
void push_ (
```

std::list< Type > & list,
const Type & object)

Помещает объект в контейнер std::list.

Объект помещается в контейнер std::list при этом контейнер остается отсортированным.

Аргументы

list	
object	

```
246                                                     {
247
248     auto it_pos = list.begin();
249     // std::list<Type>::iterator it_pos = list.begin();
250
251     while ( it_pos != list.end() ) {
252         // >= лучше чем >, т.к. при > вставка сильно замедляется,
253         // если все объекты в списке равны между собой
254         if ( *it_pos >= object ) { break; }
255         it_pos++;
256     }
257
258     list.insert(it_pos, object);
259 }
```