Programovacie techniky

Pretažovanie operátorov v C++

Vladislav Novák

Obsah

```
operator[]          a[5]
operator*           * it
operator++          it ++
operator!=          a != b
operator bool()     if (a)    bool(a)
```

`explicit` - umožnuje len explicitné pretypovanie (zakazuje implicitné pretypovanie)

Preťažovanie operátorov

Triedy

main:

```
Array array(5);
size_t size = array.getSize();
int value = array.get(0);
array.set(0, 100);
```

Preťažovanie operátorov

Triedy

```
class Array {




};
```

main:

```
Array array(5);
size_t size = array.getSize();
int value = array.get(0);
array.set(0, 100);
```

Preťažovanie operátorov

Triedy

```
class Array {
private:
    int *data;
    const size_t size;




};
```

main:

```
Array array(5);
size_t size = array.getSize();
int value = array.get(0);
array.set(0, 100);
```

Preťažovanie operátorov

Triedy

```
class Array {
private:
    int *data;
    const size_t size;
public:
    explicit Array(size_t size);
    ~Array();
    size_t getSize() const noexcept;
    int get(size_t index) const;
    void set(size_t index, int value);
};
```

main:

```
Array array(5);
size_t size = array.getSize();
int value = array.get(0);
array.set(0, 100);
```

Preťažovanie operátorov

Triedy

```
class Array {
private:
    int *data;
    const size_t size;
public:
    explicit Array(size_t size);
    ~Array();
    size_t getSize() const noexcept;
    // ?
};
```

main:

```
Array array(5);
size_t size = array.getSize();
int value = array[0];
```

Preťažovanie operátorov

Triedy

```
class Array {
private:
   int *data;
   const size_t size;
public:
   explicit Array(size_t size);
   ~Array();
   size_t getSize() const noexcept;
   int   operator[](size_t index);
};
```

main:

```
Array array(5);
size_t size = array.getSize();
int value = array[0];
```

Preťažovanie operátorov

Triedy

```
class Array {
private:
   int *data;
   const size_t size;
public:
   explicit Array(size_t size);
   ~Array();
   size_t getSize() const noexcept;
   int & operator[](size_t index);
};
```

main:

```
Array array(5);
size_t size = array.getSize();
int value = array[0];
array[0] = 100;
```

Preťažovanie operátorov

Triedy

```
class Array {
private:
   int *data;
   const size_t size;
public:
   explicit Array(size_t size);
   ~Array();
   size_t getSize() const noexcept;
   int & operator[](size_t index);
};
```

```
Array::Array(size_t size) : size(size) {
   data = new int [size];
}
```

main:

```
Array array(5);
size_t size = array.getSize();
int value = array[0];
array[0] = 100;
```

Preťažovanie operátorov

Triedy

```
class Array {
private:
   int *data;
   const size_t size;
public:
   explicit Array(size_t size);
   ~Array();
   size_t getSize() const noexcept;
   int & operator[](size_t index);
};
```

```
Array::Array(size_t size) : size(size) {
   data = new int [size];
}
Array::~Array() {
   delete[] data;
}
```

main:

```
Array array(5);
size_t size = array.getSize();
int value = array[0];
array[0] = 100;
```

Preťažovanie operátorov

Triedy

```
class Array {
private:
   int *data;
   const size_t size;
public:
   explicit Array(size_t size);
   ~Array();
   size_t getSize() const noexcept;
   int & operator[](size_t index);
};
```

```
Array::Array(size_t size) : size(size) {
   data = new int [size];
}
Array::~Array() {
   delete[] data;
}
size_t Array::getSize() const noexcept {
   return size;
}
```

main:

```
Array array(5);
size_t size = array.getSize();
int value = array[0];
array[0] = 100;
```

Preťažovanie operátorov

Triedy

```
class Array {
private:
   int *data;
   const size_t size;
public:
   explicit Array(size_t size);
   ~Array();
   size_t getSize() const noexcept;
   int & operator[](size_t index);
};
```

```
Array::Array(size_t size) : size(size) {
   data = new int [size];
}
Array::~Array() {
   delete[] data;
}
size_t Array::getSize() const noexcept {
   return size;
}
int & Array::operator[](size_t index) {
   if(index < size) {
      return data[index];
   }
   else {
      throw out_of_range(to_string(index));
   }
}
```

main:

```
Array array(5);
size_t size = array.getSize();
int value = array[0];
array[0] = 100;
```

Preťažovanie operátorov

Triedy

```
class Array {
private:
   int *data;
   const size_t size;
public:
   explicit Array(size_t size);
   ~Array();
   size_t getSize() const noexcept;
   int & operator[](size_t index);
};
```

main:

Preťažovanie operátorov (iterátor)

Triedy

```
class Array {
private:
    int *data;
    const size_t size;
public:
    explicit Array(size_t size);
    ~Array();
    size_t getSize() const noexcept;
    int & operator[](size_t index);

    // ?
};
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array {
private:
    int *data;
    const size_t size;
public:
    explicit Array(size_t size);
    ~Array();
    size_t getSize() const noexcept;
    int & operator[](size_t index);

    // ?
};
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array {
private:
    int *data;
    const size_t size;
public:
    explicit Array(size_t size);
    ~Array();
    size_t getSize() const noexcept;
    int & operator[](size_t index);

    class Iterator;
    // ?
};
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array {
private:
    int *data;
    const size_t size;
public:
    explicit Array(size_t size);
    ~Array();
    size_t getSize() const noexcept;
    int & operator[](size_t index);

    class Iterator;
    Iterator begin();
    Iterator end();
};
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array {
private:
    int *data;
    const size_t size;
public:
    explicit Array(size_t size);
    ~Array();
    size_t getSize() const noexcept;
    int & operator[](size_t index);

    class Iterator;
    Iterator begin();
    Iterator end();
};
```

```
Array::Iterator Array::begin() {
    return Iterator(this);
}
Array::Iterator Array::end() {
    return Iterator(this, size);
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
Array::Iterator Array::begin() {
    return Iterator(this);
}
Array::Iterator Array::end() {
    return Iterator(this, size);
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;




};
```

```
Array::Iterator Array::begin() {
    return Iterator(this);
}
Array::Iterator Array::end() {
    return Iterator(this, size);
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;
public:
    explicit Iterator(Array * array, size_t index = 0);




};
```

```
Array::Iterator Array::begin() {
    return Iterator(this);
}
Array::Iterator Array::end() {
    return Iterator(this, size);
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;
public:
    explicit Iterator(Array * array, size_t index = 0);




};
```

```
Array::Iterator::Iterator(Array * array, size_t index)
    : array(array)
    , index(index) {
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;
public:
    explicit Iterator(Array * array, size_t index = 0);
    int & operator*() const;




};
```

```
Array::Iterator::Iterator(Array * array, size_t index)
    : array(array)
    , index(index) {
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;
public:
    explicit Iterator(Array * array, size_t index = 0);
    int & operator*() const;




};
```

```
Array::Iterator::Iterator(Array * array, size_t index)
    : array(array)
    , index(index) {
}
int & Array::Iterator::operator*() const {
    return (*array)[index];
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;
public:
    explicit Iterator(Array * array, size_t index = 0);
    int & operator*() const;
    Iterator & operator++() noexcept; // prefixovy



};
```

```
Array::Iterator::Iterator(Array * array, size_t index)
    : array(array)
    , index(index) {
}
int & Array::Iterator::operator*() const {
    return (*array)[index];
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;
public:
    explicit Iterator(Array * array, size_t index = 0);
    int & operator*() const;
    Iterator & operator++() noexcept; // prefixovy



};
```

```
Array::Iterator::Iterator(Array * array, size_t index)
    : array(array)
    , index(index) {
}
int & Array::Iterator::operator*() const {
    return (*array)[index];
}
Array::Iterator & Array::Iterator::operator++() noexcept {
    ++ index;
    return *this;
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;
public:
    explicit Iterator(Array * array, size_t index = 0);
    int & operator*() const;
    Iterator & operator++() noexcept; // prefixovy
  //Iterator operator++(int) noexcept; // postfixovy


};
```

```
Array::Iterator::Iterator(Array * array, size_t index)
    : array(array)
    , index(index) {
}
int & Array::Iterator::operator*() const {
    return (*array)[index];
}
Array::Iterator & Array::Iterator::operator++() noexcept {
    ++ index;
    return *this;
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;
public:
    explicit Iterator(Array * array, size_t index = 0);
    int & operator*() const;
    Iterator & operator++() noexcept; // prefixovy
  //Iterator operator++(int) noexcept; // postfixovy


};
```

```
Array::Iterator::Iterator(Array * array, size_t index)
    : array(array)
    , index(index) {
}
int & Array::Iterator::operator*() const {
    return (*array)[index];
}
Array::Iterator & Array::Iterator::operator++() noexcept {
    ++ index;
    return *this;
}
//Array::Iterator Array::Iterator::operator++(int) noexcept {
//    Iterator previous(*this);
//    operator++();
//    return previous;
//}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;
public:
    explicit Iterator(Array * array, size_t index = 0);
    int & operator*() const;
    Iterator & operator++() noexcept; // prefixovy
  //Iterator operator++(int) noexcept; // postfixovy


};
```

```
Array::Iterator::Iterator(Array * array, size_t index)
    : array(array)
    , index(index) {
}
int & Array::Iterator::operator*() const {
    return (*array)[index];
}
Array::Iterator & Array::Iterator::operator++() noexcept {
    ++ index;
    return *this;
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;
public:
    explicit Iterator(Array * array, size_t index = 0);
    int & operator*() const;
    Iterator & operator++() noexcept; // prefixovy
  //Iterator operator++(int) noexcept; // postfixovy
    bool operator!=(const Iterator &other) const
                                          noexcept;
};
```

```
Array::Iterator::Iterator(Array * array, size_t index)
    : array(array)
    , index(index) {
}
int & Array::Iterator::operator*() const {
    return (*array)[index];
}
Array::Iterator & Array::Iterator::operator++() noexcept {
    ++ index;
    return *this;
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (iterátor)

Triedy

```
class Array::Iterator {
private:
    Array *array;
    size_t index;
public:
    explicit Iterator(Array * array, size_t index = 0);
    int & operator*() const;
    Iterator & operator++() noexcept; // prefixovy
  //Iterator operator++(int) noexcept; // postfixovy
    bool operator!=(const Iterator &other) const
                                          noexcept;
};
```

```
Array::Iterator::Iterator(Array * array, size_t index)
    : array(array)
    , index(index) {
}
int & Array::Iterator::operator*() const {
    return (*array)[index];
}
Array::Iterator & Array::Iterator::operator++() noexcept {
    ++ index;
    return *this;
}
bool Array::Iterator::operator!=(const Array::Iterator &other) const
                                                    noexcept {
    return this->array != other.array || this->index != other.index;
}
```

main:

```
Array array(5);

for(Array::Iterator it = array.begin(); it != array.end(); ++it) {
    cout << (*it) << endl;
}
```

```
Array array(5);

for(int element : array) {
    cout << element << endl;
}
```

Preťažovanie operátorov (súbor)

Triedy

main:

```
File file("output.txt");

if(file) {
    // .....
}
```

Preťažovanie operátorov (súbor)

Triedy

main:

```
File file("output.txt");

if(! file) {
    // .....
}
```

Preťažovanie operátorov (súbor)

Triedy

```
class File {




};
```

main:

```
File file("output.txt");

if(! file) {
    // .....
}
```

Preťažovanie operátorov (súbor)

Triedy

```
class File {
private:
   const string name;




};
```

main:

```
File file("output.txt");

if(! file) {
   // .....
}
```

Preťažovanie operátorov (súbor)

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);


};
```

main:

```
File file("output.txt");

if(! file) {
   // .....
}
```

Preťažovanie operátorov (súbor)

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);


};
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
```

main:

```
File file("output.txt");

if(! file) {
   // .....
}
```

Preťažovanie operátorov (súbor)

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;

};
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
```

main:

```
File file("output.txt");

if(! file) {
   // .....
}
```

Preťažovanie operátorov (súbor)

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;

};
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}
```

main:

```
File file("output.txt");

if(! file) {
   // .....
}
```

Preťažovanie operátorov (súbor)

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}
```

main:

```
File file("output.txt");

if(! file) {
   // .....
}
```

explicit

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}
```

main:

```
File file("output.txt");

if(! file) { // v podmienke môžeme použiť aj s deklaráciou operátora ako explicit
   // .....
}
```

explicit

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}
```

main:

```
File file("output.txt");

if(file) { // v podmienke môžeme použiť aj s deklaráciou operátora ako explicit
   // .....
}
```

explicit

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}
```

main:

```
File file("output.txt");
```

explicit

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}
```

main:

```
File file("output.txt");

bool value = file; // chyba, musime explicitne pretypovat, alebo vymazat explicit
```

explicit

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}
```

main:

```
File file("output.txt");

bool value = bool(file); // ok
```

explicit

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};

void fn(bool value);
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}

void fn(bool value) {
   // .....
}
```

main:

```
File file("output.txt");

bool value = bool(file); // ok
```

explicit

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};

void fn(bool value);
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}

void fn(bool value) {
   // .....
}
```

main:

```
File file("output.txt");

bool value = bool(file); // ok

fn(file); // chyba, musime explicitne pretypovat, alebo vymazat explicit
```

explicit

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};

void fn(bool value);
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}

void fn(bool value) {
   // .....
}
```

main:

```
File file("output.txt");

bool value = bool(file); // ok

fn(bool(file)); // ok
```

explicit

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};

void fn(bool value);
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}

void fn(bool value) {
   // .....
}
```

main:

explicit

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};

void fn(bool value);
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}

void fn(bool value) {
   // .....
}
```

main:

```
File file1 = string("output1.txt"); // chyba, kompilátor nerobí explicitné pretypovanie
```

explicit

Triedy

```
class File {
private:
   const string name;
public:
   explicit File(const string & name);
   explicit operator bool() const noexcept;
   // ďalšie metódy, deštruktor
};

void fn(bool value);
```

```
File::File(const string & name) : name(name) {
    /* otvorenie suboru */
}
File::operator bool() const noexcept {
   return true/false;
}

void fn(bool value) {
   // .....
}
```

main:

```
File file1 = string("output1.txt"); // chyba, kompilátor nerobí explicitné pretypovanie

string name2 = "output2.txt";
File file2 = name2; // chyba, kompilátor nerobí explicitné pretypovanie
```