C++ CoreHard Spring 2018

25 years of C++ history flashed in front of my eyes

Yauheni Akhotnikau

stiffstream

A few words about myself

Started programming somewhere in 1990.

C++ is the main tool since 1992 with a few short pauses.

Was a developer, team-lead, head of software development department.

Now a co-founder of small software startup.

A dictator behind OpenSource projects SObjectizer and RESTinio.

Why I'm doing this talk?

Some know me as a "C++ advocate". It's because:

don't bite the hand feeding you.

I've done some investments in C++ and want to protect them.

There are too many non-technical obstacles like myths and urban legends.

The question of life, universe and...

How and why
C++ became popular?

Fast growth of popularity

500 users in 1985

400 000 users in 1991

3 000 000+ users at the end of 1990s.

"According to IDC, out of 13.1 million developer seats worldwide in 1999, Visual Basic had 7.2 million vs. 3.6 million C/C++ seats and 1.3 million Java seats."

quote from Application Development Trends magazine (2001.05.29)

Why?

Three main factors

(very subjective)

OOP was a "silver bullet"

There is no "silver bullet" but every time something pretends to be it.

In 1980s it was OOP.

OOP is great technique and significantly simplifies some tasks.

Easy upgrade from plain C

Significant part of C is a subset of C++.

Powerful features of C++ can be added to a C program only when needed.

Existing C libraries can be easily used from C++.

Computers were weak

Jan 1987. IBM PS/2 Model 70: 16MHz CPU, 6MiB RAM.

Oct 1988. The first NeXT computer: 25MHz CPU, 8MiB RAM.

Jan 1989. DECstation 3100: 16.7MHz CPU, from 4MiB to 24MiB RAM.

Apr 1989. SPARCstation 1: 20MHz CPU, from 4MiB to 64MiB RAM.

http://pctimeline.info/workstation/work1987.htm,
https://en.wikipedia.org/wiki/DECstation,
https://en.wikipedia.org/wiki/SPARCstation_1,
http://www.computinghistory.org.uk/det/2586/ibm-ps-2-model-70/

As a consequence...

There was a strong need for:

- a highly expressive and powerful;
- a very efficient;
- a practical

programming language for masses.

C++ become that language.

And another important factor...

C++ in mid 1980s was significantly simpler than modern C++ ;)

Some examples from my own experience

How C++ caught myself

My way in early 1990s

Basic ⇒ Pascal ⇒ Turbo Pascal ⇒ C ⇒ C++

Pascal has such useful thing as ***set***.

There wasn't such thing in C. And there wasn't a way to implement it.

There wasn't such thing in C++ (at 1991).

But there was a way to implement it.

My way in early 1990s

Basic ⇒ Pascal

Pascal has such

There wasn't su

There wasn't su

But there was a

```
class int_set {
  int * items_;
  ...
public:
  int_set() : items_(new int[initial_capacity]) {...}
  ~int_set() { delete[] items_; }
  ...
};

int_set operator+(const int_set & a, const int_set & b) {...}
int_set operator-(const int_set & a, const int_set & b) {...}
int_set operator&(const int_set & a, const int_set & b) {...}

int_set received_items;
int_set checked_items;
...
int_set delayed_items = received_items - checked_items;
```

stiffstream

Linear algebra computations

There was a task of solving a system of linear equations.

A very special case: symmetric band matrix.

$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$	0	0	0	0
	0	0	0	0	0	...	
$X_{1,2}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$	$X_{2,5}$	0	0	0
	0	0	0	0	0	...	
$X_{1,3}$	$X_{2,3}$	$X_{3,3}$	$X_{3,4}$	$X_{3,5}$	$X_{3,6}$	0	0
	0	0	0	0	0	...	
$X_{1,4}$	$X_{2,4}$	$X_{3,4}$	$X_{4,4}$	$X_{4,5}$	$X_{4,6}$	$X_{4,7}$	0
	0	0	0	0	0	...	
0	$X_{2,5}$	$X_{3,5}$	$X_{4,5}$	$X_{5,5}$	$X_{5,6}$	$X_{5,7}$	
	$X_{5,8}$	0	0	0	0	0	
	...						
0	0	$X_{3,6}$	$X_{4,6}$	$X_{5,6}$	$X_{6,6}$	$X_{6,7}$	
	$X_{6,8}$	$X_{6,9}$	0	0	0	0	

stiffstream

Linea

There w

A very sp

```
class BandMatrix;

class RowAccessor {
  BandMatrix & matrix;
  unsigned row_index;
public:
  RowAccessor(BandMatrix & m, unsigned i);
  float & operator[](unsigned j) {...}
};

class BandMatrix {
  float * data;
public
  BandMatrix(unsigned dimension, unsigned band_width) {...};
  ...
  RowAccessor operator[](unsigned i) { return RowAccessor(*this, i); }
  ...
};

BandMatrix m;
...
m[i][j] = m[i][j-1] + m[i-1][j] + ...;
```

$X_{1,}$

$X_{1,}$

$X_{1,}$

$X_{1,}$

0

0 0 $X_{3,6}$ $X_{4,6}$ $X_{5,6}$ $X_{6,6}$ $X_{6,7}$

$X_{6,8}$ $X_{6,9}$ 0 0 0 0

A bit of history...

30+ years of C++ evolution

in one slide

Just a brief overview

1980s: official release, very quick evolution.

1990s: become a mainstream, the first ISO standard (C++98), lot of incompatible libraries.

2000s: still a mainstream, but the oblivion is started. C++03. Long-term construction C++0x. Boost libraries. LLVM and clang.

2010s: a new hope. C++11, C++14, C++17. C++ Core Guidelines. C++20 on its way...

Yet more history...

Some important notes about 1990s

1990s: life without the standard

A lot of incompatible "basic" libraries (some of them supplied with compilers).

Every big and complex C++ project has its own "standard" library.

There could be several independent implementation of String class in one project.

1990s: rising of C++ templates

STL shows the real power of C++ templates.

Java 1.0 in 1995 demonstrated how to "live" without templates/generics.

Template metaprogramming was accidentally discovered in 1994.
(http://aszt.inf.elte.hu/~gsd/halado_cpp/ch06s04.html#Static-metaprogramming)

As consequence: Alexandrescu's "Modern C++ Design" in 2001.

C++ will never be the same anymore ;)

A picture is worth a thousand words

Show me the code!

C++ evolution in simple examples

By the help of a simple demo application:

- there is a collection of `struct place` (each struct contains name, price and rating);
- sort this collection by rating, then by price, then by name. And then print the collection after sorting;
- sort this collection by name, then print it.

Ancient C++ (pre-C++98)

```
struct place {
  char * name_;
  long price_;
  int rating_;

  place() : name_(0) {}
  place(const char * name, long price, int rating)
     : name_(copy_name(name)), price_(price), rating_(rating) {}
  place(const place & o)
    : name_(copy_name(o.name_)), price_(o.price_)
    , rating_(o.rating_) {}
  ~place() { delete[] name_; }

  place & operator=(const place & o) {
     char * copy = copy_name(o.name_);
     delete[] name_;
     name_ = copy; price_ = o.price_; rating_ = o.rating_;
     return *this;
  }
```

```
  static char * copy_name(const char * src) {
     char * r = 0;
     if(src) {
       r = new char[strlen(src) + 1];
       strcpy(r, src);
     }
     return r;
  }
};
```

Ancient C++ (pre-C++98)

```
ostream & operator<<(ostream & to, const place & p) {
  to << setw(20) << left << p.name_ << "[" << p.rating_ << "]"
     << setw(7) << right << p.price_;
  return to;
}

void print(const place * places, size_t count) {
  for(size_t i = 0; i != count; ++i)
     cout << places[i] << endl;
  cout << endl;
}
```

Ancient C++ (pre-C++98)

```
int compare_by_rating_price_name(const void * av, const void * bv) {
  const place & a = *(const place *)av;
  const place & b = *(const place *)bv;
  int result;
  if(a.rating_ < b.rating_) result = -1;
  else {
    if(a.rating_ == b.rating_) {
      if(a.price_ < b.price_) result = -1;
      else {
        if(a.price_ == b.price_) result = strcmp(a.name_, b.name_);
        else result = 1;
      }
    }
    else result = 1;
  }
  return result;
}
```

Ancient C++ (pre-C++98)

```
int compare_by_name(const void * av, const void * bv) {
  const place & a = *(const place *)av;
  const place & b = *(const place *)bv;

  return strcmp(a.name_, b.name_);
}
```

Ancient C++ (pre-C++98)

```
void get_places(place *& places, size_t & count) {
  places = new place[7];
  places[0] = place("Moscow", 1500, 3);
  places[1] = place("St. Petersburg", 1300, 5);
  places[2] = place("Minsk", 500, 4);
  places[3] = place("Tokyo", 3500, 4);
  places[4] = place("Sydney", 5000, 3);
  places[5] = place("Paris", 3000, 5);
  places[6] = place("London", 3500, 4);
  count = 7;
}
```

Ancient C++ (pre-C++98)

```
int main() {
  place * places;
  size_t count;

  get_places(places, count);

  qsort(places, count, sizeof(place), compare_by_rating_price_name);
  print(places, count);

  qsort(places, count, sizeof(place), compare_by_name);
  print(places, count);

  delete[] places;
}
```

Old C++ (C++98/03)

```
using namespace std;

struct place {
  string name_;
  long price_;
  int rating_;

  place() {}
  place(const char * name, float price, int rating)
     : name_(name), price_(price), rating_(rating) {}
};
```

Old C++ (C++98/03)

```
ostream & operator<<(ostream & to, const place & p) {
  to << setw(20) << left << p.name_ << "[" << p.rating_ << "]"
    << setw(7) << right << p.price_;
  return to;
}

template<class C>
void print(const C & places) {
  for(typename C::const_iterator i = places.begin(); i != places.end(); ++i)
    cout << *i << endl;
  cout << endl;
}
```

Old C++ (C++98/03)

```
bool compare_by_rating_price_name(const place & a, const place & b) {
  bool result = false;
  if(a.rating_ < b.rating_) result = true;
  else if(a.rating_ == b.rating_) {
    if(a.price_ < b.price_) result = true;
    else if(a.price_ == b.price_) result = (a.name_ < b.name_);
  }
  return result;
}

bool compare_by_name(const place & a, const place & b) {
  return a.name_ < b.name_;
}
```

Old C++ (C++98/03)

```
vector<place> get_places() {
  vector<place> places;
  places.push_back(place("Moscow", 1500, 3));
  places.push_back(place("St. Petersburg", 1300, 5));
  places.push_back(place("Minsk", 500, 4));
  places.push_back(place("Tokyo", 3500, 4));
  places.push_back(place("Sydney", 5000, 3));
  places.push_back(place("Paris", 3000, 5));
  places.push_back(place("London", 3500, 4));
  return places;
}
```

Old C++ (C++98/03)

```
int main() {
  vector<place> places = get_places();

  sort(places.begin(), places.end(), compare_by_rating_price_name);
  print(places);

  sort(places.begin(), places.end(), compare_by_name);
  print(places);
}
```

stiffstream

Modern C++ (C++17)

```
using namespace std;

struct place {
  string name_;
  long price_;
  int rating_;
};
```

Modern C++ (C++17)

```
ostream & operator<<(ostream & to, const place & p) {
  to << setw(20) << left << p.name_ << "[" << p.rating_ << "]"
    << setw(7) << right << p.price_;
  return to;
}

template<class C>
void print(const C & places) {
  for(const auto & p : places)
    cout << p << endl;
  cout << endl;
}
```

Modern C++ (C++17)

```
bool compare_by_rating_price_name(const place & a, const place & b) {
  return tie(a.rating_, a.price_, a.name_) < tie(b.rating_, b.price_, b.name_);
}

bool compare_by_name(const place & a, const place & b) {
  return a.name_ < b.name_;
}
```

Modern C++ (C++17)

```
vector<place> get_places() {
 return {
   {"Moscow", 1500, 3},
   {"St. Petersburg", 1300, 5},
   {"Minsk", 500, 4},
   {"Tokyo", 3500, 4},
   {"Sydney", 5000, 3},
   {"Paris", 3000, 5},
   {"London", 3500, 4}
 };
}
```

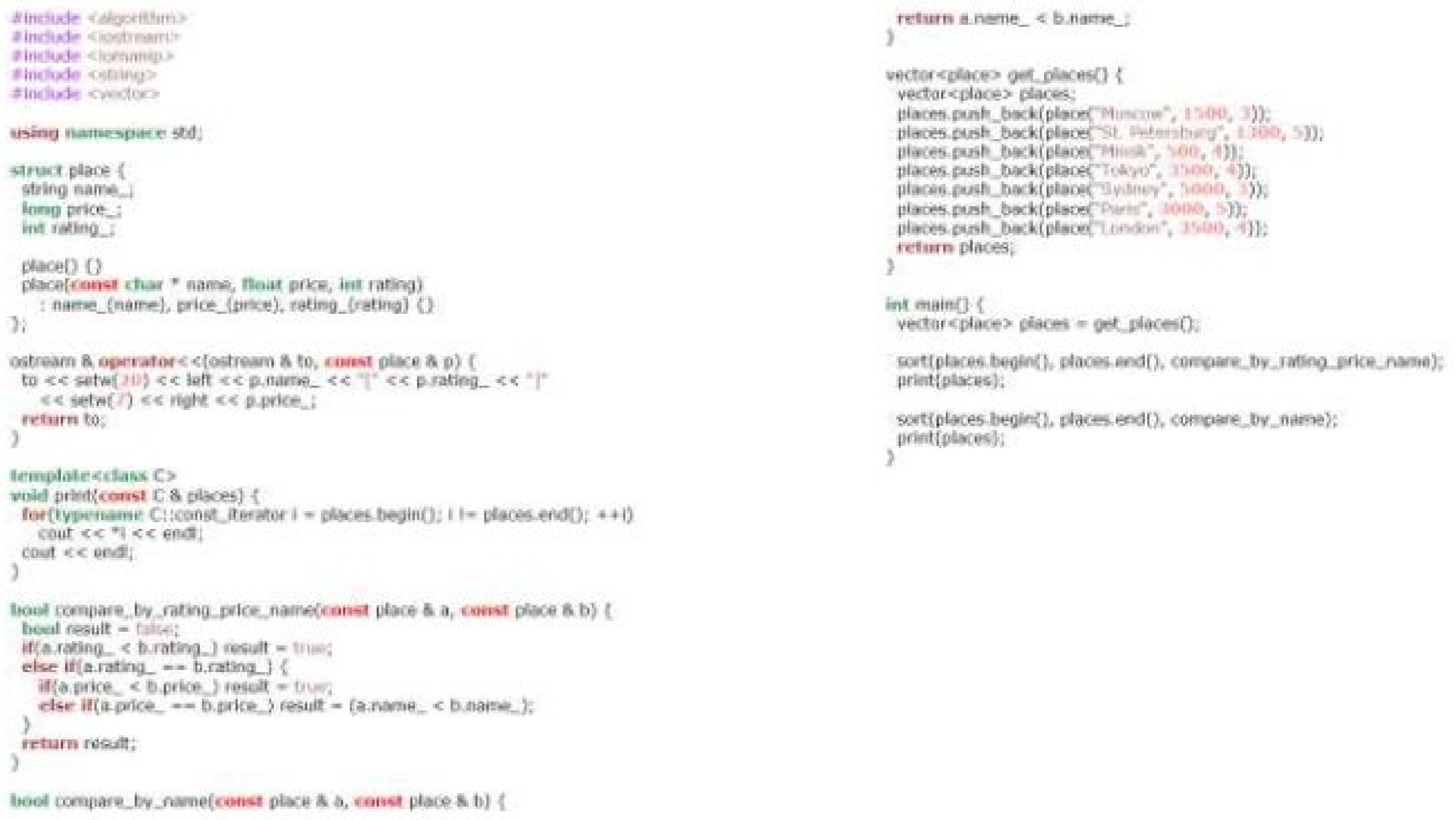
Modern C++ (C++17)

```
int main() {
  auto places = get_places();

  sort(begin(places), end(places), compare_by_rating_price_name);
  print(places);

  sort(begin(places), end(places), compare_by_name);
  print(places);
}
```

Full text of pre-C++98 example

```cpp
#include <iostream.h>
#include <iomanip.h>
#include <string.h>
#include <stdlib.h>

struct place {
  char * name_;
  long price_;
  int rating_;

  place() : name_(0) {}
  place(const char * name, long price, int rating)
    : name_(copy_name(name)), price_(price), rating_(rating)
{}
  place(const place & o)
    : name_(copy_name(o.name_)), price_(o.price_),
rating_(o.rating_) {}
  ~place() {
    delete[] name_;
  }

  place & operator=(const place & o) {
    char * copy = copy_name(o.name_);
    delete[] name_;
    name_ = copy;
    price_ = o.price_;
    rating_ = o.rating_;
    return *this;
  }

  static char * copy_name(const char * src) {
    char * r = 0;
    if(src) {
    r = new char[strlen(src) + 1];
    strcpy(r, src);
    }
    return r;
  }
};

ostream & operator<<(ostream & to, const place & p) {
  to << setw(20) << left << p.name_ << "[" << p.rating_ <<
"]"
    << setw(7) << right << p.price_;
  return to;
}

void print(const place * places, size_t count) {
  for(size_t i = 0; i != count; ++i)
    cout << places[i] << endl;
  cout << endl;
}

int compare_by_rating_price_name(const void * av, const
void * bv) {
  const place & a = *(const place *)av;
  const place & b = *(const place *)bv;

  int result;
  if(a.rating_ < b.rating_) result = -1;
  else {
    if(a.rating_ == b.rating_) {
      if(a.price_ < b.price_) result = -1;
      else {
        if(a.price_ == b.price_) {
        result = strcmp(a.name_, b.name_);
        }
        else result = 1;
    }
    }
    else result = 1;
  }

  return result;
}

int compare_by_name(const void * av, const void * bv) {
  const place & a = *(const place *)av;
  const place & b = *(const place *)bv;

  return strcmp(a.name_, b.name_);
}

void get_places(place *& places, size_t & count) {
  places = new place[7];
  places[0] = place("Moscow", 1500, 3);
  places[1] = place("St. Petersburg", 1300, 5);
  places[2] = place("Minsk", 500, 4);
  places[3] = place("Tokyo", 3500, 4);
  places[4] = place("Sydney", 5000, 3);
  places[5] = place("Paris", 3000, 5);
  places[6] = place("London", 3500, 4);
  count = 7;
}

int main() {
  place * places;
  size_t count;

  get_places(places, count);

  qsort(places, count, sizeof(place),
compare_by_rating_price_name);
  print(places, count);

  qsort(places, count, sizeof(place), compare_by_name);
  print(places, count);

  delete[] places;
}
```

Full text of C++98 example

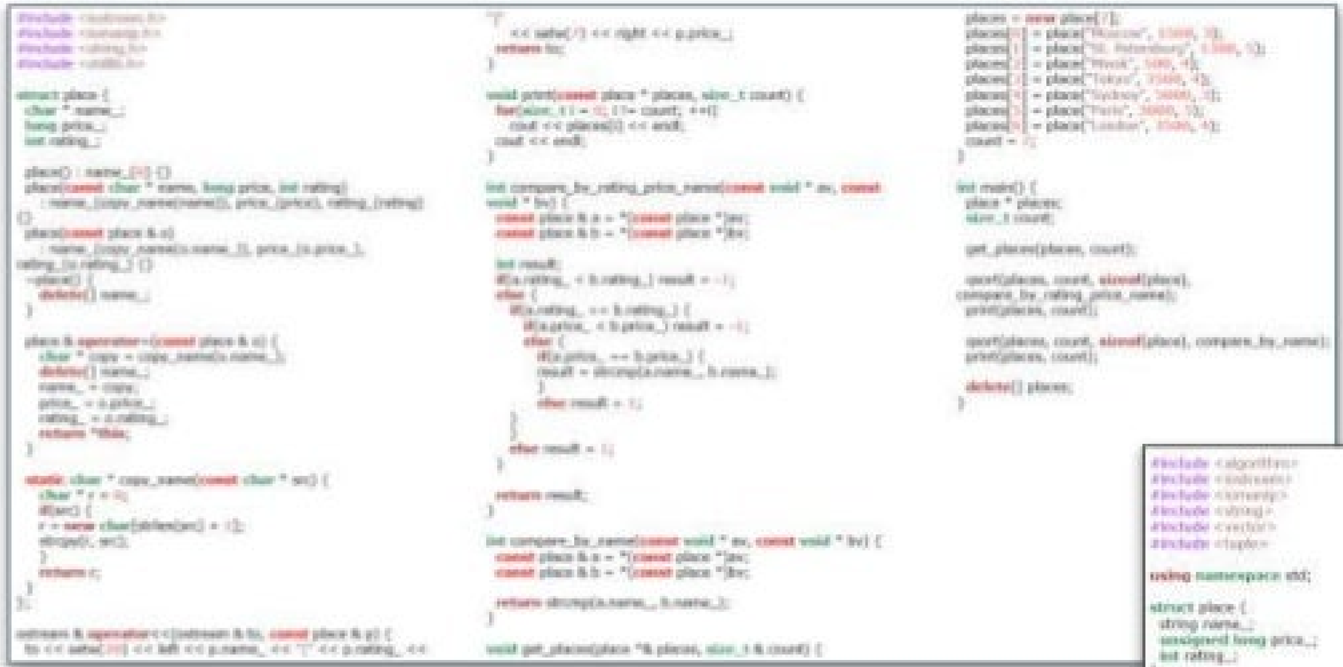


```cpp
#include <algorithm>
#include <iostream>
#include <iomanip>
#include <string>
#include <vector>

using namespace std;

struct place {
  string name_;
  long price_;
  int rating_;

  place() {}
  place(const char * name, float price, int rating)
    : name_(name), price_(price), rating_(rating) {}
};

ostream & operator<<(ostream & to, const place & p) {
  to << setw(20) << left << p.name_ << "[" << p.rating_ << "]"
    << setw(7) << right << p.price_;
  return to;
}

template<class C>
void print(const C & places) {
  for(typename C::const_iterator i = places.begin(); i != places.end(); ++i)
    cout << *i << endl;
  cout << endl;
}

bool compare_by_rating_price_name(const place & a, const place & b) {
  bool result = false;
  if(a.rating_ < b.rating_) result = true;
  else if(a.rating_ == b.rating_) {
    if(a.price_ < b.price_) result = true;
    else if(a.price_ == b.price_) result = (a.name_ < b.name_);
  }
  return result;
}

bool compare_by_name(const place & a, const place & b) {
  return a.name_ < b.name_;
}

vector<place> get_places() {
  vector<place> places;
  places.push_back(place("Moscow", 1500, 3));
  places.push_back(place("St. Petersburg", 1300, 5));
  places.push_back(place("Minsk", 500, 4));
  places.push_back(place("Tokyo", 3500, 4));
  places.push_back(place("Sydney", 5000, 3));
  places.push_back(place("Paris", 3000, 5));
  places.push_back(place("London", 3500, 4));
  return places;
}

int main() {
  vector<place> places = get_places();

  sort(places.begin(), places.end(), compare_by_rating_price_name);
  print(places);

  sort(places.begin(), places.end(), compare_by_name);
  print(places);
}
```

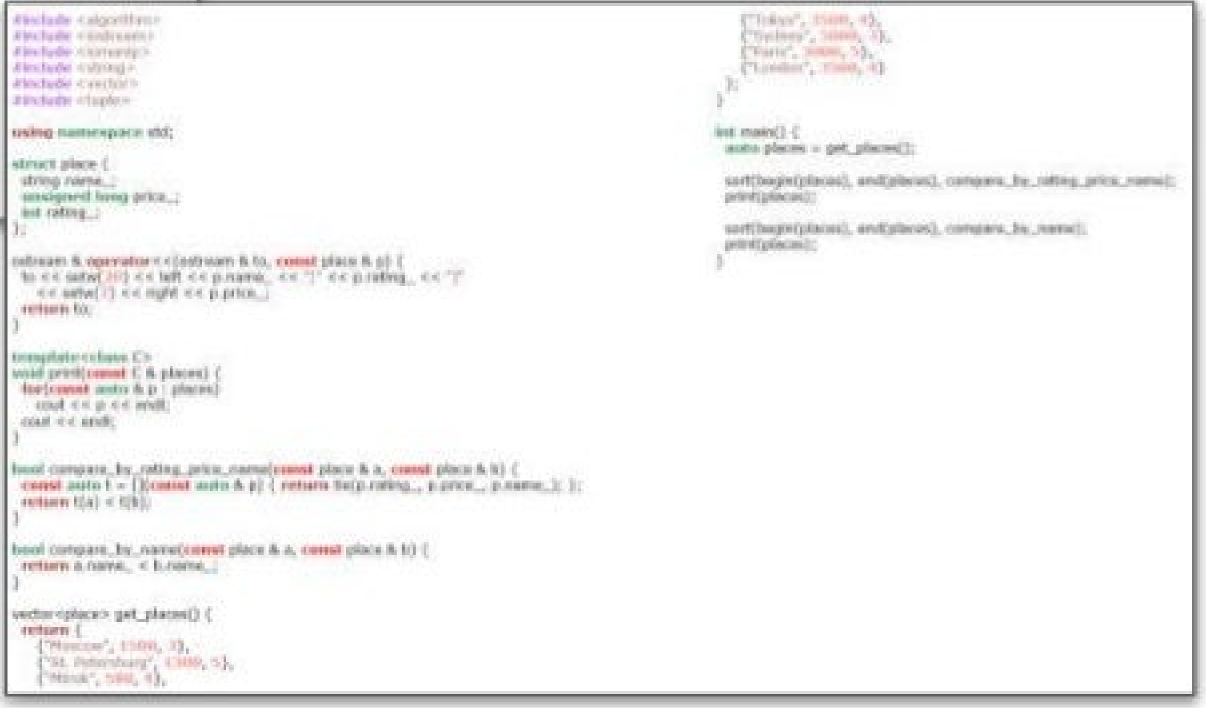
Full text of C++17 example

```cpp
#include <algorithm>
#include <iostream>
#include <iomanip>
#include <string>
#include <vector>
#include <tuple>

using namespace std;

struct place {
  string name_;
  unsigned long price_;
  int rating_;
};

ostream & operator<<(ostream & to, const place & p) {
  to << setw(20) << left << p.name_ << "[" << p.rating_ << "]"
     << setw(7) << right << p.price_;
  return to;
}

template<class C>
void print(const C & places) {
  for(const auto & p : places)
    cout << p << endl;
  cout << endl;
}

bool compare_by_rating_price_name(const place & a, const place & b) {
  const auto t = [](const auto & p) { return tie(p.rating_, p.price_, p.name_); };
  return t(a) < t(b);
}

bool compare_by_name(const place & a, const place & b) {
  return a.name_ < b.name_;
}

vector<place> get_places() {
  return {
    {"Moscow", 1500, 3},
    {"St. Petersburg", 1300, 5},
    {"Minsk", 500, 4},
    {"Tokyo", 3500, 4},
    {"Sydney", 5000, 3},
    {"Paris", 3000, 5},
    {"London", 3500, 4}
  };
}

int main() {
  auto places = get_places();

  sort(begin(places), end(places), compare_by_rating_price_name);
  print(places);

  sort(begin(places), end(places), compare_by_name);
  print(places);
}
```

30 years of C++ evolution in a picture



```cpp
#include <iostream.h>
#include <iomanip.h>
#include <string.h>
#include <stdlib.h>

struct place {
  char * name_;
  long price_;
  int rating_;

  place() : name_(0) {}
  place(const char * name, long price, int rating)
    : name_(copy_name(name)), price_(price), rating_(rating)
{}
  place(const place & o)
    : name_(copy_name(o.name_)), price_(o.price_),
rating_(o.rating_) {}
  ~place() {
    delete[] name_;
  }

  place & operator=(const place & o) {
    char * copy = copy_name(o.name_);
    delete[] name_;
    name_ = copy;
    price_ = o.price_;
    rating_ = o.rating_;
    return *this;
  }

  static char * copy_name(const char * src) {
    char * r = 0;
    if(src) {
    r = new char[strlen(src) + 1];
    strcpy(r, src);
    }
    return r;
  }
};

ostream & operator<<(ostream & to, const place & p) {
  to << setw(20) << left << p.name_ << "[" << p.rating_ << 
"]"
    << setw(7) << right << p.price_;
  return to;
}

void print(const place * places, size_t count) {
  for(size_t i = 0; i != count; ++i)
    cout << places[i] << endl;
  cout << endl;
}

int compare_by_rating_price_name(const void * av, const
void * bv) {
  const place & a = *(const place *)av;
  const place & b = *(const place *)bv;

  int result;
  if(a.rating_ < b.rating_) result = -1;
  else {
    if(a.rating_ == b.rating_) {
      if(a.price_ < b.price_) result = -1;
      else {
        if(a.price_ == b.price_) {
        result = strcmp(a.name_, b.name_);
        }
        else result = 1;
      }
    }
    else result = 1;
  }

  return result;
}

int compare_by_name(const void * av, const void * bv) {
  const place & a = *(const place *)av;
  const place & b = *(const place *)bv;

  return strcmp(a.name_, b.name_);
}

void get_places(place *& places, size_t & count) {
  places = new place[7];
  places[0] = place("Moscow", 1500, 3);
  places[1] = place("St. Petersburg", 1300, 5);
  places[2] = place("Minsk", 500, 4);
  places[3] = place("Tokyo", 3500, 4);
  places[4] = place("Sydney", 5000, 3);
  places[5] = place("Paris", 3000, 5);
  places[6] = place("London", 3500, 4);
  count = 7;
}

int main() {
  place * places;
  size_t count;

  get_places(places, count);

  qsort(places, count, sizeof(place),
compare_by_rating_price_name);
  print(places, count);

  qsort(places, count, sizeof(place), compare_by_name);
  print(places, count);

  delete[] places;
}
```



```cpp
#include <algorithm>
#include <iostream>
#include <iomanip>
#include <string>
#include <vector>
#include <tuple>

using namespace std;

struct place {
  string name_;
  unsigned long price_;
  int rating_;
};

ostream & operator<<(ostream & to, const place & p) {
  to << setw(20) << left << p.name_ << "[" << p.rating_ << "]"
    << setw(7) << right << p.price_;
  return to;
}

template<class C>
void print(const C & places) {
  for(const auto & p : places)
    cout << p << endl;
  cout << endl;
}

bool compare_by_rating_price_name(const place & a, const place & b) {
  const auto t = [](const auto & p) { return tie(p.rating_, p.price_, p.name_); };
  return t(a) < t(b);
}

bool compare_by_name(const place & a, const place & b) {
  return a.name_ < b.name_;
}

vector<place> get_places() {
  return {
    {"Moscow", 1500, 3},
    {"St. Petersburg", 1300, 5},
    {"Minsk", 500, 4},
    {"Tokyo", 3500, 4},
    {"Sydney", 5000, 3},
    {"Paris", 3000, 5},
    {"London", 3500, 4}
  };
}

int main() {
  auto places = get_places();

  sort(begin(places), end(places), compare_by_rating_price_name);
  print(places);

  sort(begin(places), end(places), compare_by_name);
  print(places);
}
```

stiffstream

Nothing can last forever

How C++ lost its popularity?

Computers become strong

1997-1999. Cheap PCs with:

- Intel or AMD processors at 200, 266, 300, 350 and more MHz;
- dozens or even hundreds of MiB of RAMs.

10x growth of execution speed in 10 years.

New slogan: "CPU and memory is not a resource".

As a result: *Java is not slow anymore*

(you can use name of any safe language with GC instead of Java: Python, SmallTalk, Scheme and so on).

Programmer's productivity matters

C++ is not safe:

- no run-time checks;
- a lot of UB.

C++ requires an attention for a lot of details.

Manual memory management vs Garbage collection.

It's hard to support C++ in IDE.

No batteries included. No dependency manager. Incompatible libraries.

New "hot" niches

1980s and early 1990s: desktop & GUI applications.

Late 1990s and early 2000s: Internet/WWW + server-side.

Late 2000s and 2010s: mobile.

Free lunch is over or...

Is there a place for C++ today?

C++ is (and always was) a niche language

The niche for C++ is a software systems which are:

- complex enough to require a powerful and high level language and
- have strict requirements to performance and/or resource consumption.

Yet another feature of this niche:

- there is no place for quick-and-dirty solutions!

Where C++ can be used now?

High performance computing (weather forecasting, molecular modeling, physic simulation, big-data, machine learning, ...)

Low resource consumption apps (mobile application, internet of things, ...)

System programming and around (OS kernels, drivers and utilities, middleware, compilers, antiviruses, ...)

Real-time and embedded systems (SCADA, industrial internet of things, avionics, automotive, finance, telecom, defense, ...)

Complex desktop applications (CADs, photoshop, AAA games, browsers, office apps, ...)

Cross-platform apps (app's kernel in C++ + GUI in native technology (Dropbox's jinie), ...)

Another important question

Why C++'s case is so unique?

There are not many similar examples

C++ is one of the few languages that:

- has many years history of history;
- has an international standard and several live and evolving compilers;
- has a legacy of enormous size;
- is still widely used;
- is still under the active development;
- has no single owner (or just big corporation behind it).

I can remember only Cobol, FORTRAN and Ada.

What does it mean?

You can't just stop using C++ and rewrite all C++ code in another modern language.

You can't just throw out any part of the language.

You can't leave just one C++ compiler and kill all the other C++ compilers.

You can't force all C++ developers to use one "right" code style.

You can't force all C++ developers to use one set of "right" tools (like libraries, dependency managers or build tools).

Just remember:

C++ is community driven language

That's all!

Thank you!

Yauheni Akhotnikau, StiffStream

https://stiffstream.com

eao197@stiffstream.com