Силата и слабостта на C++

Стефан Чеканов

PlovDev
12 октомври 2013
Пловдив

За лектора

- Обича да прави софтуер
- Прави го от много отдавна
- Програмира на C/C++, Java, PHP, C#, JavaScript, SQL, HTML, CSS, shell, assembly, Word, Excel и каквото още дойде
- Правил е много неща, включително Java Virtual Machine (JVM)
- Основното му занимание в момента е

BROSIX

www.brosix.com

Защо C++?

- Защо да се занимаваме с него?

  Нали го изучаваме в университет (училище).

  А и има много по-интересни (и лесни) езици от него.

Фокусиран (ограничен) поглед

Кое е по-добро?

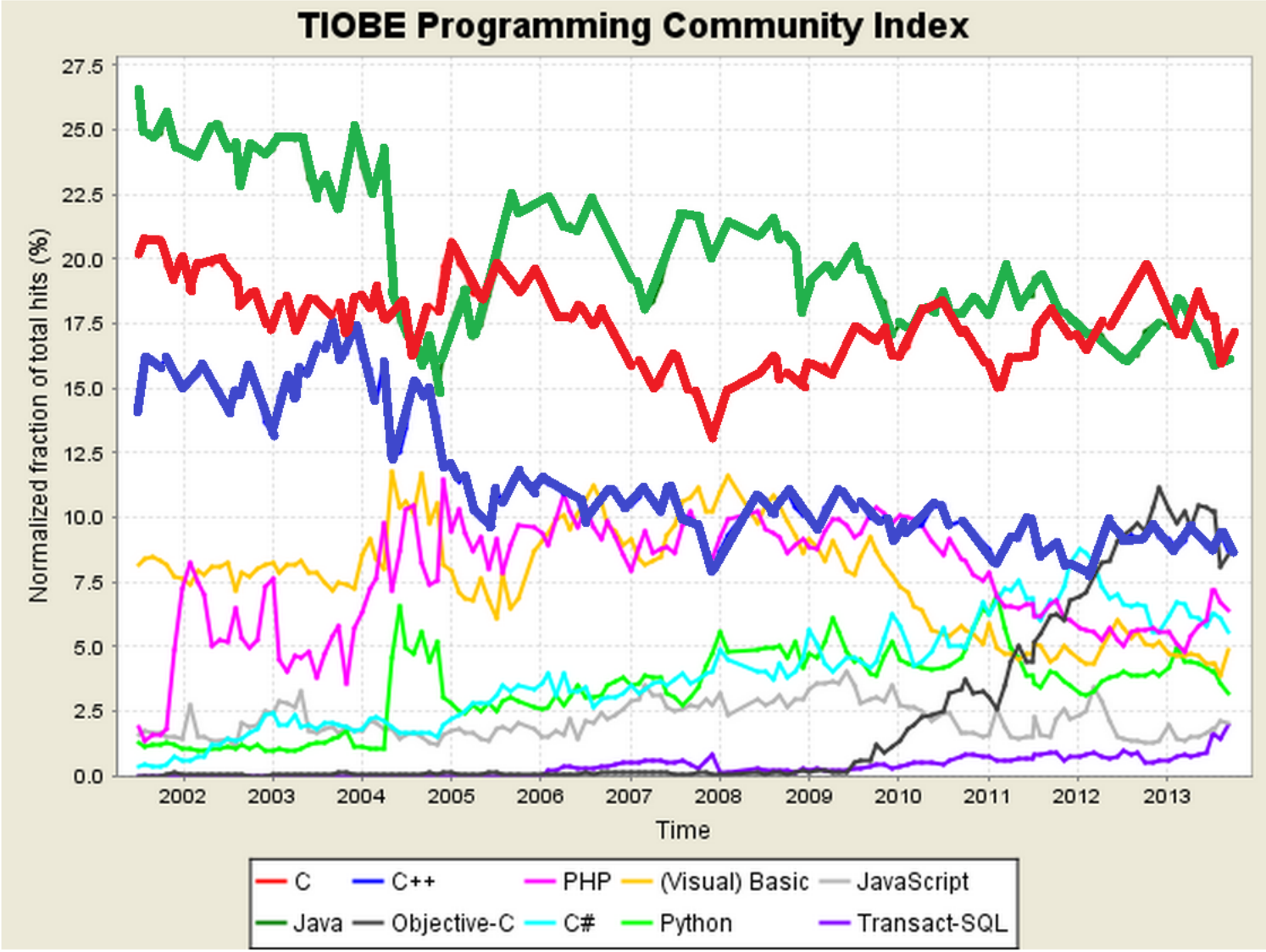
Малко статистика

- **TIOBE Programming Community Index for September 2013**
- Индикатор за популярността на програмните езици
- Обновява се веднъж месечно от 25 години
- Базира се на броя професионалисти в световен мащаб, курсове и доставчици
- TIOBE index НЕ показва кой е *най-добрият* език за програмиране или на кой са написани *най-много линии код*

TIOBE Programming Community Index for September 2013

Position Sep 2013	Position Sep 2012	Delta in Position	Programming Language	Ratings Sep 2013	Delta Sep 2012	Status
1	1	=	C	16.975%	-2.32%	A
2	2	=	Java	16.154%	-0.11%	A
3	4	↑	C++	8.664%	-0.48%	A
4	3	↓	Objective-C	8.561%	-1.21%	A
5	6	↑	PHP	6.430%	+0.82%	A
6	5	↓	C#	5.564%	-1.03%	A
7	7	=	(Visual) Basic	4.837%	-0.69%	A
8	8	=	Python	3.169%	-0.69%	A
9	11	↑↑	JavaScript	2.015%	+0.69%	A
10	14	↑↑↑↑	Transact-SQL	1.997%	+1.12%	A

Дългосрочна статистика



Общата картина –Позиции на топ 10 езиците преди 5, 15 и 25 години

Programming Language	Position Sep 2013	Position Sep 2008	Position Sep 1998	Position Sep 1988
C	1	2	1	1
Java	2	1	4	-
C++	3	3	2	3
Objective-C	4	42	-	-
PHP	5	5	-	-
C#	6	8	-	-
(Visual) Basic	7	4	3	7
Python	8	6	28	-
JavaScript	9	9	32	-
Transact-SQL	10	29	-	-
Lisp	16	16	13	2

ЗАЩО ?

Защо е толкова напред в индекса и за толкова дълго време?

Силата на C++

- да бъде с вас!

Силата на C++

- Много-платформи – използва се на почти всички платформи – Windows, Linux, Mac, Android, iOS, Solaris, FreeBSD, NetBSD, VxWorks, embedded devices

Има и други претендиращи за “много платформеност”

Има и други претендиращи за “много платформеност”

Но за да пуснеш програма на C/C++, на компютъра на крайния потребител, на която и да е платформа, не се изисква допълнително оборудване като: виртуални машини, интерпретатори, frameworks.

- **Какво ще използвате, ако искате да направите софтуер, който да работи на Windows, Linux и Mac?**

Като казвам «софтуер» имам предвид:

- комерсиален софтуер
- софтуер, който има GUI
- софтуер, който ще се интегрира добре в ОС, т.е. ще ползва и предоставя услуги в ОС
- софтуер, който ще работи добре (бързо с малко памет)

Отговорът е: C/C++

- На практика няма друга алтернатива

Пример от практиката

Силата на C++

- Много-платформи

- **Компилира се до native код – бързодействие на програмите**

На практика е единственият съвременен език, който по подразбиране се компилира до native код

На практика е единственият съвременен език, който по подразбиране се компилира до native код

Всъщност има и други, които се компилират: Pascal (Delphi), Fortran, Ada, Cobol. Но тяхната употреба е по-ограничена, поради наличието им на ограничен набор от платформи и използването им за конкретни цели.

Други езици като Java, PHP, C# имат JIT и оптимизатори

За тях можем да си говорим много

Силата на C++

- Много-платформеност
- Компилира се до native код

- **Богат набор от библиотеки за различни приложения**

Важни библиотеки в съвременната индустрия задължително имат версия за C/C++

- Maths, VoIP, SIP, Audio, Video, Communication, GUI

Силата на C++

- Много-платформеност
- Компилира се до native код
- Богат набор от библиотеки за различни приложения

- **Memory management – управление на паметта**

Memory management

- Да навлезем в дълбоки води

Memory Management

Накои хора, като чуят, че сами трябва да си управляват паметта, изпадат в паника, страх, шок или.... просто се отказват

Не е страшно

Ако знаеш как да го използваш

Който задели паметта, той трябва да я освободи

- Закон номер 1
  на Memory Management-а

При добър код на C++, почти не се налага сам да управляваш паметта

- Закон номер 2
  на Memory Management-а

Управлението на паметта е едно от най-мощните оръжия в програмирането въобще

- Закон номер 3
  на Memory Management-а

Примерен код

```
82
83  //  This implements IDBQueryListener interface
84  void    LoadReporter::onQuery( QueryData* queryData )
85  {
86      XMLItem xml;
87      DBObject::parseQueryResult( queryData, xml );
88      int numberOfUsers   = Convert::str2int( queryData->m_params["users"] );
89
90      int errCode = xml["status"].asInt();
91      if( errCode != ERR_SUCCESS )
92      {
93          logEvent( "Problem occured while sending capacity data!" );
94          return;
95      }
96
97      m_updateTimeout     = xml["LoadTimeout"  ].asInt();
98      m_updateThreshold   = xml["LoadThreshold"].asInt();
99
100     //  Notify the core to extract the list of servers from our response.
101     theApp->m_core.onQuery( queryData );
102
103     m_nextUpdateTime    = time(NULL) + m_updateTimeout;
104
105     logEvent( TextFormat("Capacity reported [connected users = %1] [new UTM: %
106 }
107 ////////////////////////////////////////////////////////////////////////////
```

Можем още много да говорим за Memory management, но да продължим нататък

Силата на C++

- Много-платформеност
- Компилира се до native код
- Богат набор от библиотеки за различни приложения
- Memory management

- **Конструкции на езика – неща които дават истинска сила и мощ**

Конструкции на езика

- или
  “Дълбоките тайни на Силата”

Конструкции на езика

- Макроси – наследство от C
- Разделение на декларация от реализация – .h и .cpp
- Templates (шаблони)
- Предефиниране на оператори

Конструкции на езика

- Макроси – наследство от C
- Разделение на декларация от реализация – .h и .cpp
- Templates (шаблони)
- **Предефиниране на оператори**

Предефиниране на оператори

- Позволява подмяна на стандартни оператори в езика

Предефиниране на оператори

- operator =
- operator +, -, *, /, %, &, |, ^, !
- operator ++, --
- operator +=, -=, &=, |=, *=, /=, %=, ^=
- operator ==, !=, <, <=, >, >=
- operator &&, ||
- operator >>, <<
- operator []
- operator ()
- operator ->
- operator new, delete

Примерен код

```
82
83  //  This implements IDBQueryListener interface
84  void    LoadReporter::onQuery( QueryData* queryData )
85  {
86      XMLItem xml;
87      DBObject::parseQueryResult( queryData, xml );
88      int numberOfUsers   = Convert::str2int( queryData->m_params["users"] );
89
90      int errCode = xml["status"].asInt();
91      if( errCode != ERR_SUCCESS )
92      {
93          logEvent( "Problem occured while sending capacity data!" );
94          return;
95      }
96
97      m_updateTimeout     = xml["LoadTimeout"  ].asInt();
98      m_updateThreshold   = xml["LoadThreshold"].asInt();
99
100     //  Notify the core to extract the list of servers from our response.
101     theApp->m_core.onQuery( queryData );
102
103     m_nextUpdateTime    = time(NULL) + m_updateTimeout;
104
105     logEvent( TextFormat("Capacity reported [connected users = %1] [new UTM: %
106 }
107 ////////////////////////////////////////////////////////////////////////////////
```

Предефиниране на оператори

- Позволява да се използват познати конструкции, за нови класове данни
- Увеличава се четимостта на кода
- Прави кода по-лесен за поддръжка
- В крайна сметка води до по-качествен код

Предефиниране на оператори

- operator =
- operator +, -, *, /, %, &, |, ^, !
- operator ++, --
- operator +=, -=, &=, |=, *=, /=, %=, ^=
- operator ==, !=, <, <=, >, >=
- operator &&, ||
- operator >>, <<
- operator []
- operator ()
- operator ->
- **operator new, delete**

Предефинирането на *operator new* и *operator delete*

```
281         Message* msg    = new Message();
282         user->sendMessage( msg );
283     }
```

- Позволява лесно да се реализира нова система за Memory management, без да се променя кода, където се използва класа
- Позволява писането на код оптимизиран за скорост и консумация на памет

Принцип 1 на програмирането

- Кодът не може едновременно да е бърз и да консумира малко памет
  - Ако кодът е бърз, то ще консумира много памет
  - Ако кодът консумира малко памет, ще е бавен
  - Но, кодът може да е едновременно бавен и да консумира много памет

Принцип 1 на програмирането

- Кодът не може едновременно да е бърз и да консумира малко памет
  - Ако кодът е бърз, то ще консумира много памет
  - Ако кодът консумира малко памет, ще е бавен
  - Но, кодът може да е едновременно бавен и да консумира много памет

Средствата на C++ позволяват да се наруши този основен принцип на програмирането. Или поне да се отслаби действието му.

- Много-платформеност
- Компилира се до native код
- Богат набор от библиотеки за различни приложения
- Memory management
- Конструкции на езика

Силата на C++

Истинската сила идва от умелата комбинация на всички възможности на езика

Слабостта на C++

- или
  “Как да не взривя планетата”

Слабостта на C++

- Слабостта на C++ е неговата сила

Слабостта на C++

Неправилното използване на езика превърща силните му страни в слабост

- Слабостта на C++ е неговата сила

- Изкушението да използваш не на място всичките му възможности

Слабостта на C++

Неправилното използване на езика превърща силните му страни в слабост

Вместо заключение

- или
  “За какво ми е всичко това?”

За какво ми е всичката тази мощ?

- Та аз правя уеб сайтове и приложения
- Или мобилни приложения
- А понякога някой скрипт на shell

Така си мислех и аз

- Но започнах да развивам уменията си на C/C++ въпреки ограниченията на средата в която бях:
  - Във фирмата работех на CLIPPER
  - В университета не ме обучаваха на C/C++
  - Нямах нито един приятел/познат, който да знае C/C++
  - Нямаше Интернет

Резултата

През последните 10-15 години съм работил по някои от най-интересните проекти, които могат да се случат на програмист. При това не трябваше да съм в някоя американска компания и да напускам града си.

- Java Virtual Machine
- Събиране и обработка на данни от индустриални устройства
- Voice/Video over IP – кодери, декодери, комуникация
- Screen sharing, Remote Control
- Instant Messaging

Благодаря за вниманието!

Стефан Чеканов
BROSIX

PlovDev
12 октомври 2013
Пловдив