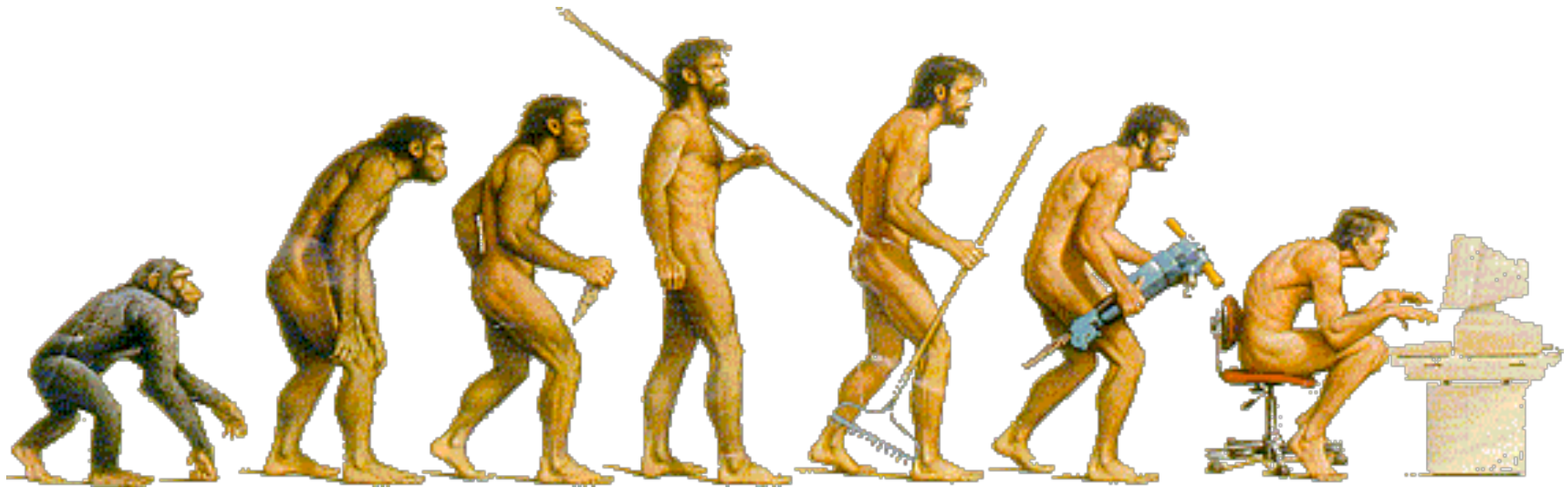
Object-Oriented Programming

Using C++



Course Contents

- Introduction to object-oriented programming…
- …with a strong software engineering foundation…
- …aimed at producing and maintaining large, high-quality software systems.

Buzzwords

responsibility-driven design

inheritance

encapsulation

iterators

overriding

coupling

cohesion

template

interface

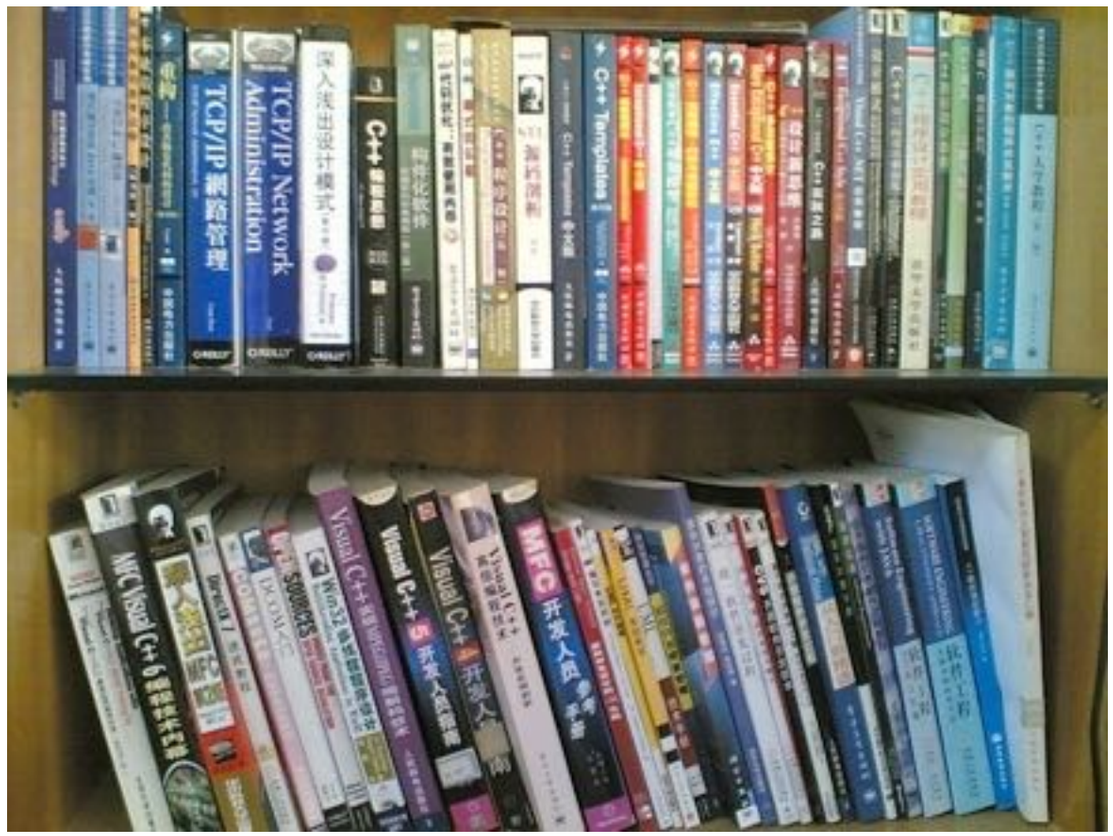
collection classes

mutator methods

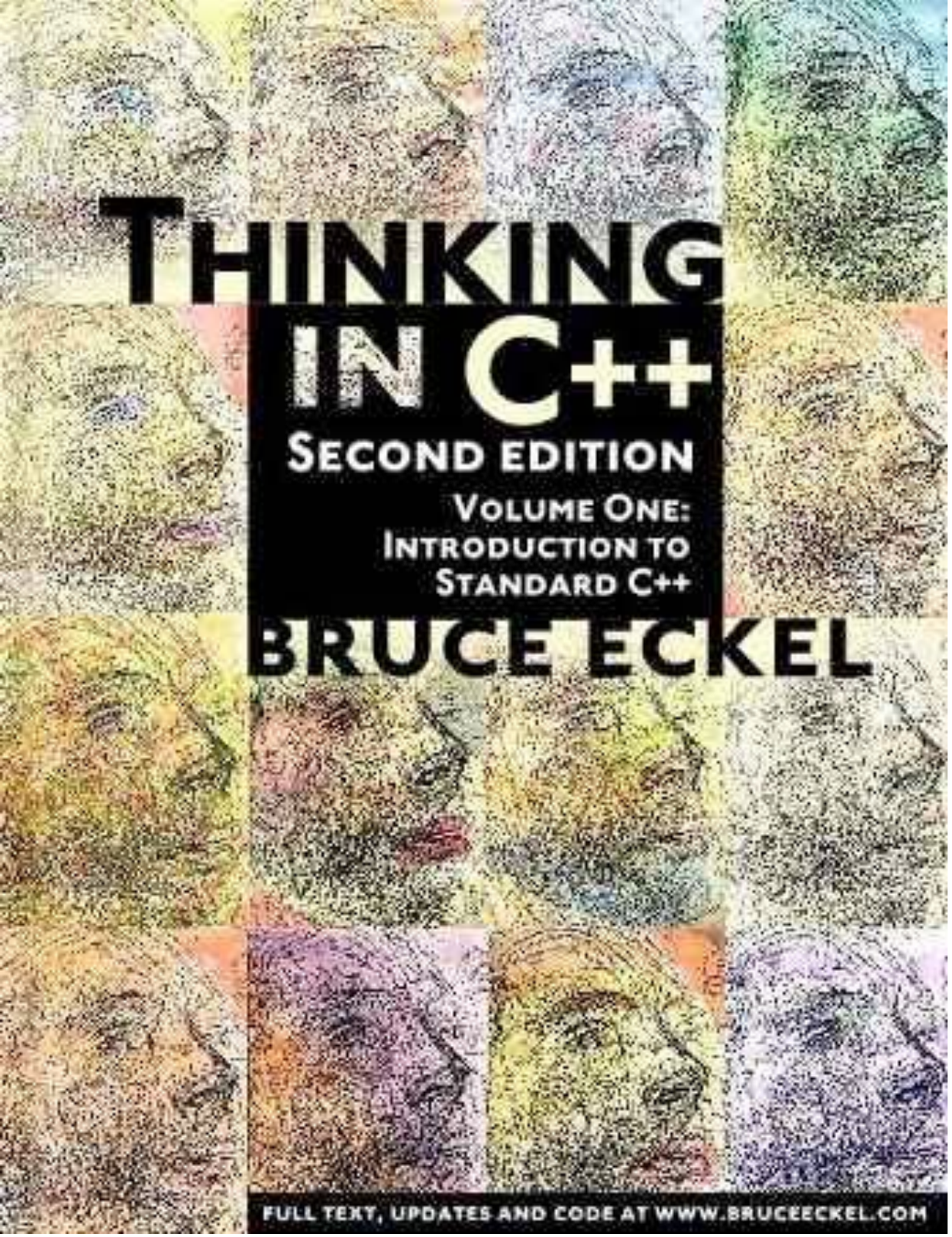
polymorphic method calls

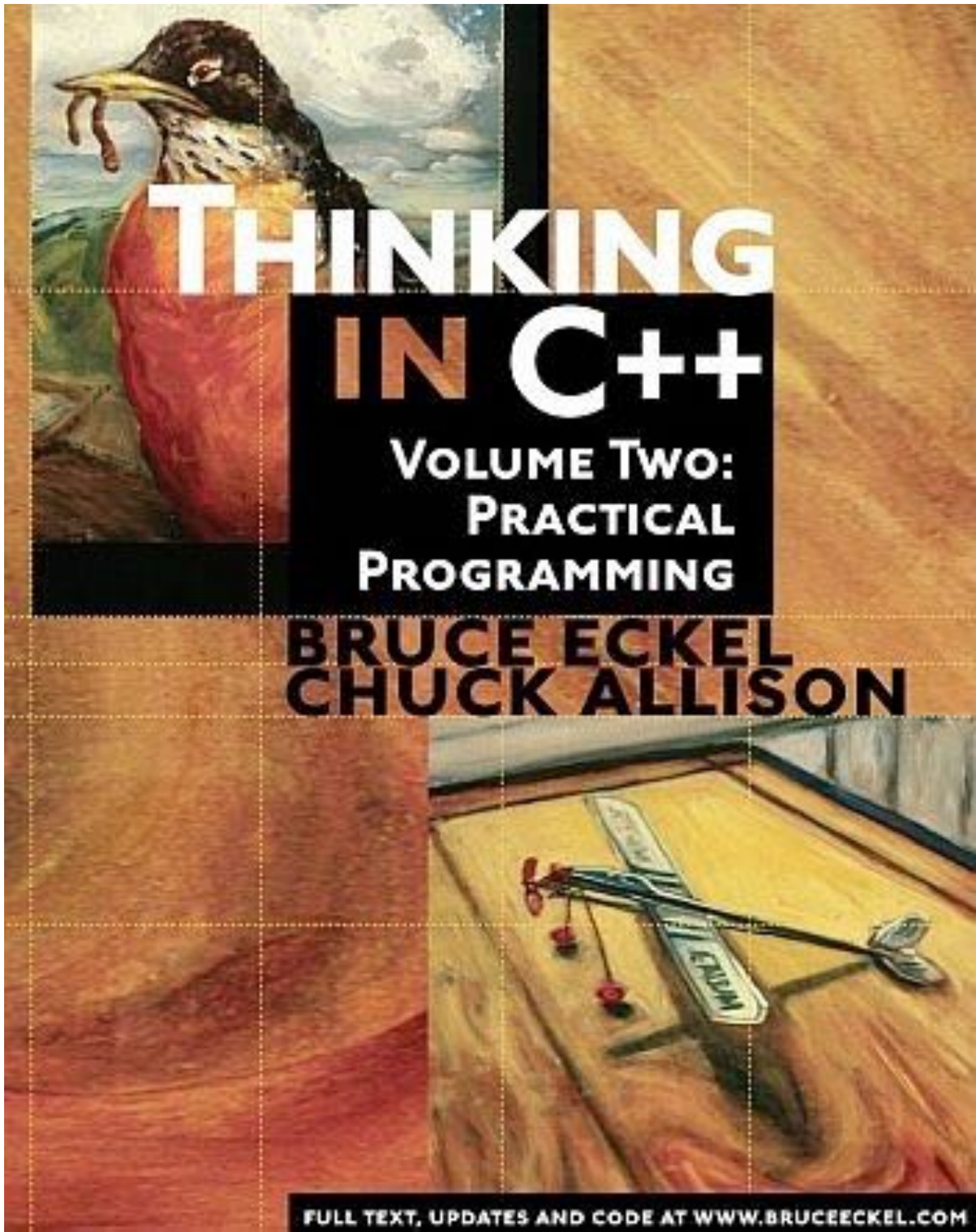
Textbooks

- Thinking In C++, Ver. 2, Vol. 1 & 2
- C++ Primer, Ver 5
- References:
  - The C++ Programming Language
  - C++: The Core Language
  - Essential C++
  - Effective C++
  - Inside the C++ Object Model
  - C++ Templates



C++编程思想 机械工业出版社





Bruce Eckel

- BRUCE ECKEL is the author of "Thinking in C++" , who won the Software Development Jolt Award for best book of 1995. He's been professionally programming for 20 years and has been teaching people throughout the world how to program with objects since 1986. He was a voting member of the C++ Standards Committee.

- http://mindview.net

PintiA

- 先注册账号，建议用@zju邮箱
  - 在个人中心输入姓名和学号绑定学号
    - 验证码：968811
- 已经绑定过就不需要再绑了

Assessment

1. Assignments: 8%, one problem set for each week, on PTA, due next lecture
2. In-class quiz: 8%, on 学在浙大 or paper work
3. 7 Labs: 14%, on PTA, during the lab periods
4. 4 Projects: 16%, on PTA, every four weeks
5. Mid-Term Exam: 4% on PTA, 90-min at lab period, week 9
6. Final Exam: 50%

Tools for C++

- TDM GCC64 (Windows) or clang+llvm (macOS)
- Visual Studio Code
- 学在浙大有安装视频
- 允许继续使用Dev C++

Introduction to C++

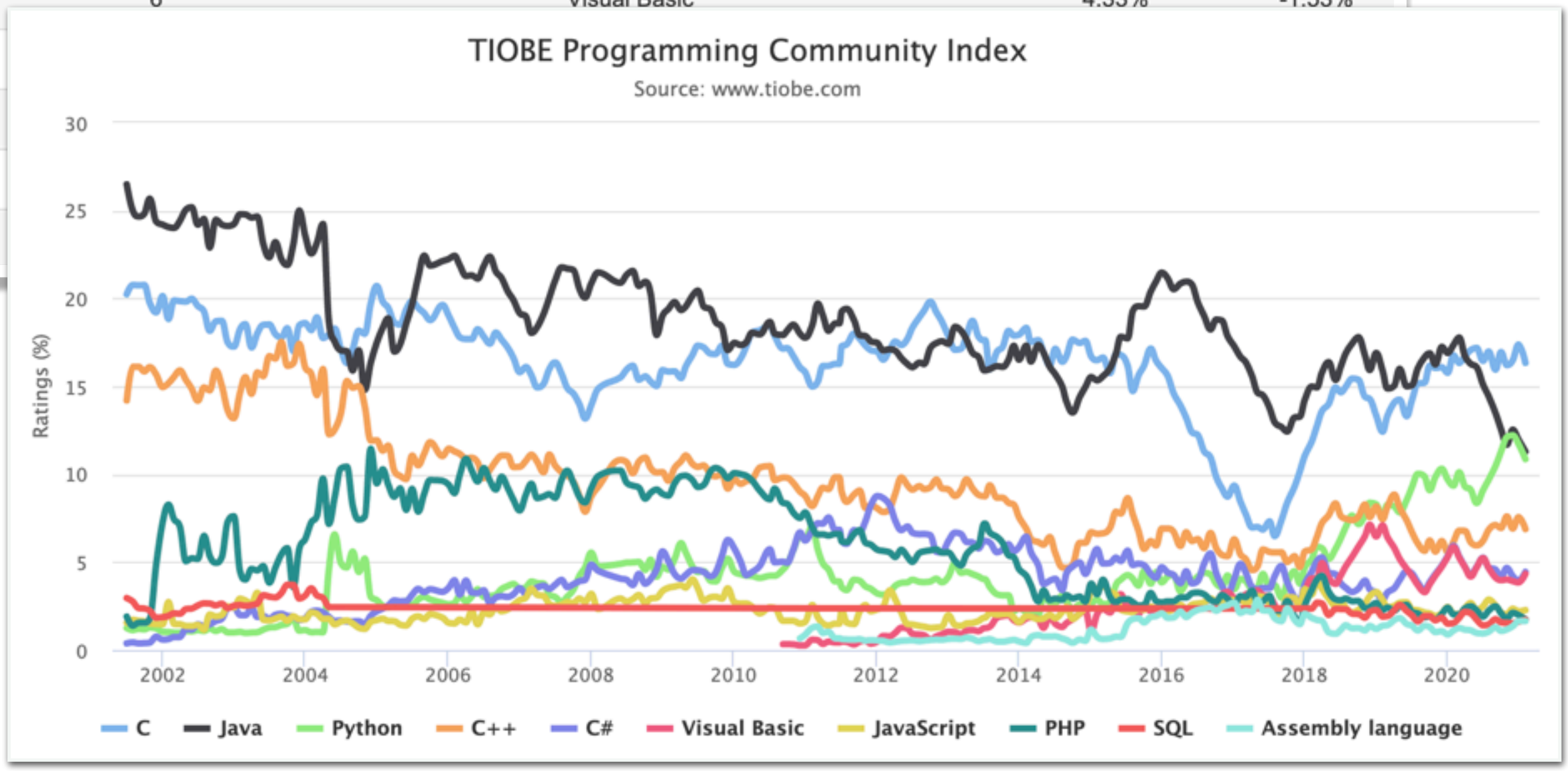
The trip begins…

学哪个语言

Feb 2021	Feb 2020	Change	Programming Language	Ratings	Change
1	2	^	C	16.34%	-0.43%
2	1	v	Java	11.29%	-6.07%
3	3		Python	10.86%	+1.52%
4	4		C++	6.88%	+0.71%
5	5		C#	4.44%	-1.48%
6	6		Visual Basic	4.33%	-1.53%
7	7		JavaScript	2.27%	+0.21%
8	8		PHP	1.75%	-0.27%
9	9		SQL	1.72%	+0.20%
10	12	^	Assembly language	1.65%	+0.54%

图片来源：http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Feb 2021	Feb 2020	Change	Programming Language	Ratings	Change
1	2	↑	C	16.34%	-0.43%
2	1	↓	Java	11.29%	-6.07%
3	3		Python	10.86%	+1.52%
4	4		C++	6.88%	+0.71%
5	5		C#	4.44%	-1.48%
6	6		Visual Basic	4.33%	-1.53%
7					
8					
9					
10					



图片来源：http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

其他语言?

```c
#include <stdio.h>

int main()
{
    printf("Hello World!\n");

    return 0;
}
```

- 现代的编程语言在语法上的差异很小
  - 几乎都是C-like语言

```java
class Hello {
    public static void main(String[] arg) {
        System.out.println("Hello World!");
    }
}
```

- 语言的能力/适用领域主要是由
  - 库, and
  - 传统所决定的

```
print "Hello World!\n"
```

结论建议

- 对计算机本身（体系结构、操作系统、编译）感兴趣—>C
- 想编程解决手头的问题（统计、AI、桌面小程序）—>Python
- 有明确的需求（求职）—>人家要什么学什么（PHP、JavaScript、C++）
- 还没想好 —>Java

学习建议

- 命令行的意识和能力
  - 编写脚本来完成工作
- 以git为代表的代码版本管理的意识和能力
- 使用第一手资料的意识和能力

Introduction to C++

The trip begins…

The C Language

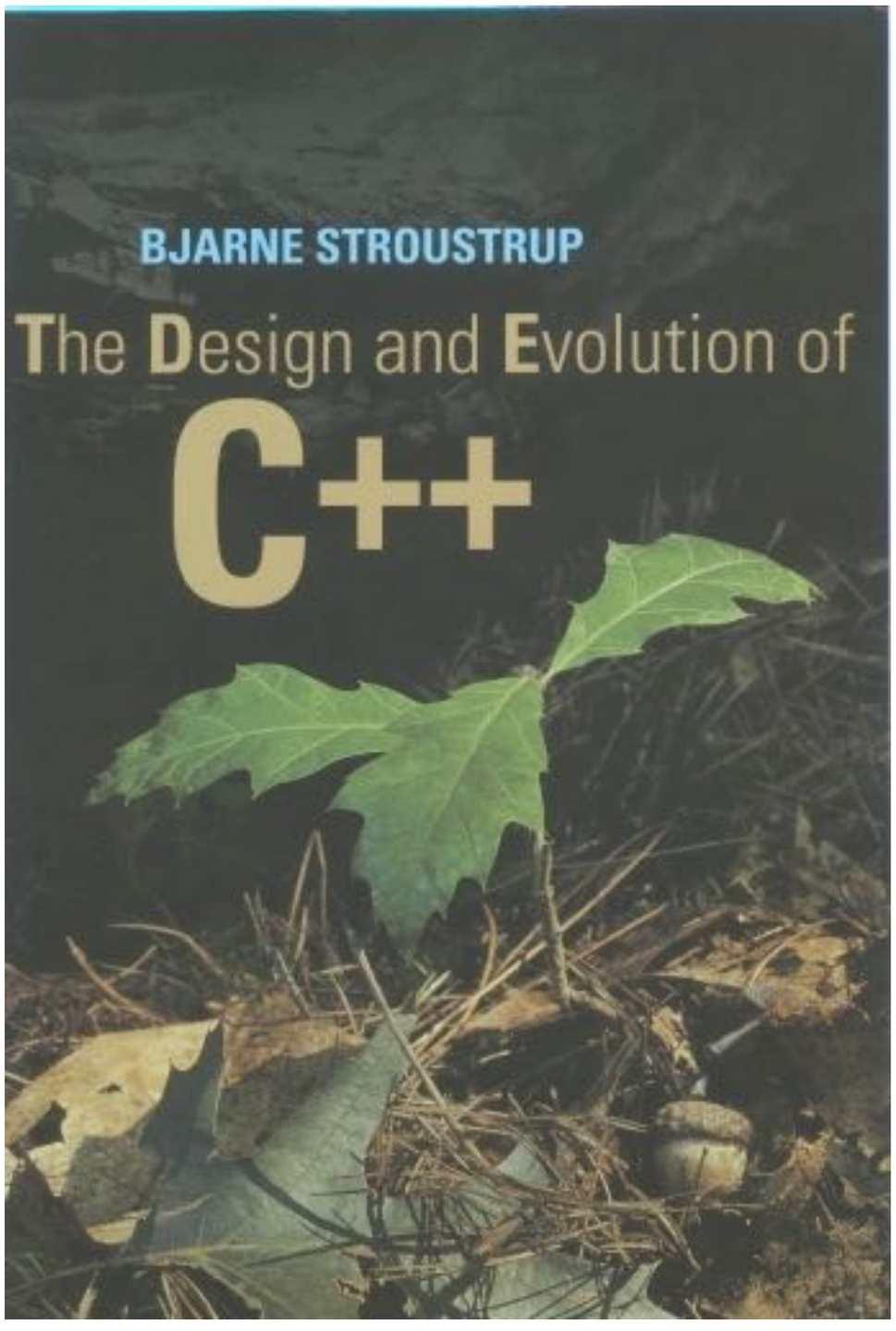
- **Strengths**
  - Efficient programs
  - Direct access to machine, suitable for OS and ES
  - Flexible
- **Weakness**
  - Insufficient type checking
  - Poor support for programming-in-the-large
  - Procedure-oriented programming

Bjarne Stroustrup

- C++ was first designed and implemented by Bjarne Stroustrup, AT&T, early 1980's
- http://www.research.att.com/~bs/homepage.html

- Oct. 2002, Stroustrup visited Zhejiang Univ.

The Design and Evolution of C++



Bjarne Stroustrup, Addison-Wesley, ISBN 0-201-54330-3

Brief history of C++ (1)

- 1978: BS at Cambridge, UK. Simulation program in Simula
- Supports classes, inheritance, and type check
- Poor performance
- http://www.engin.umd.umich.edu/CIS/course.des/cis400/simula/simula.html

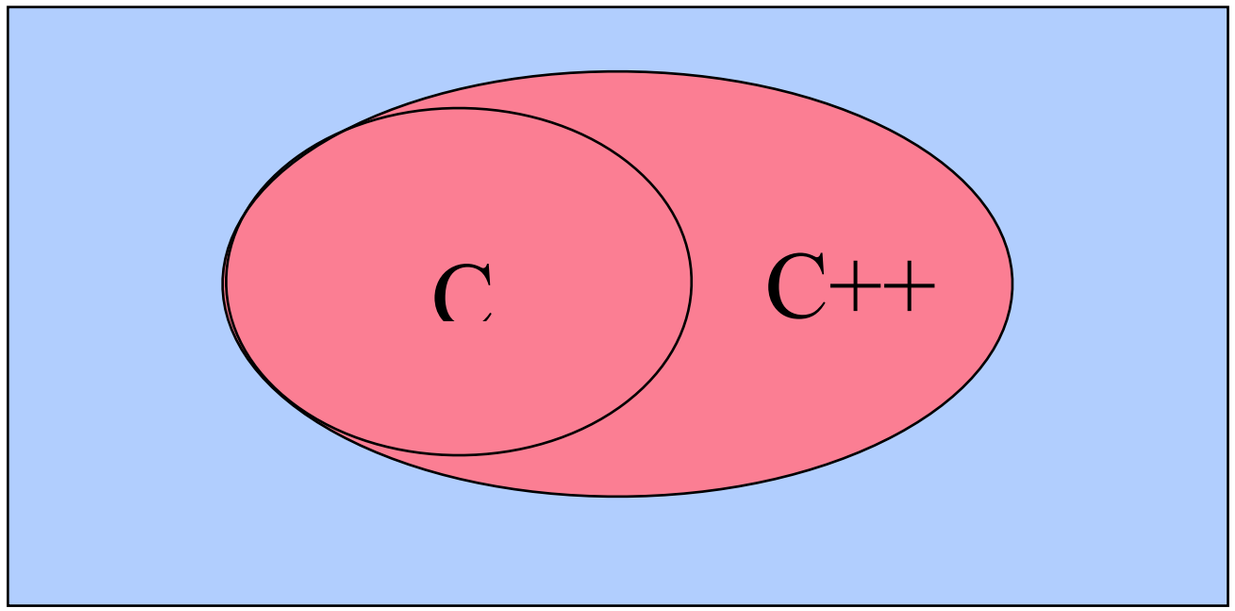
Brief history of C++ (2)

- 1979: BS at AT&T Labs, Cpre, C w/ classes
- 1980: most C++ features but virtual functions
- 1983: C++ w/ virtual functions, named C++ by Rick Mascitti
- 1985: Cfront
- 1985: "The C++ Programming Language"
- 1990: ANSI C++ Committee ISO/ANSI Standard C++ in 1998: ISO/IEC 14882 (http://www.open-std.org/jtc1/sc22/wg21/)

Goal for C++

- to combine
  - Flexibility and efficiency of C
  - Support for object oriented programming (from SmallTalk)

C and C++



- C++ builds on C
- Knowledge of C helps you in C++
- C++ support more styles of programming
- C++ provides more features

C++ improvements

- Data abstraction
- Access control
- Initialization & cleanup
- Function overloading
- Streams for I/O
- Constants (C99)
- Name control
- Inline functions(C99)
- References
- Operator overloading
- More safe and powerful memory management
- Support for OOP
- Templates
- Exception handling
- More extensive libraries, STL

C++

- C++ can be viewed as a “better” C
  - C++ => C=C+1
- but...
  - C++ is not C
    - Focus on C++ as a language in its own right
- C++ is a hybrid language, supports
  - Procedure-oriented programming
  - Object-oriented programming
  - Generic programming

The First C++ Program

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello, World! I am "
<< 18 << " Today!" << endl;

    return 0;
}
```

Read input

```
#include <iostream>
using namespace std;

int main() {
    int number;
    cout << "Enter a decimal number: ";
    cin >> number;
    cout << "The number you entered is " <<
number <<"." << endl;

    return 0;
}
```

string

The string class

- You must add this at the beginning of your code
  - `#include <string>`
- Define variable of string like other types
  - `string str;`
- Initialize it w/ string constant
  - `string str = "Hello";`
- Read and write string w/ cin/cout
  - `cin >> str;`
  - `cout << str;`

Assignment for string

```
char charr1[20];

char charr2[20] = "jaguar";

string str1;

string str2 = "panther";

carr1 = char2;    //  illegal

str1 = str2;    //  legal
```

Concatenation for string

- `string str3;`
- `str3 = str1 + str2;`
- `str1 += str2;`
- `str1 += "lalala";`

length

- s.length();
- The dot is an operator that retrieve a member of a struct in C
- It is still that operator in C++ that retrieve a member of an object in C

ctors

- string(const char *cp, int len);
- string(const string& s2, int pos);
- string(const string& s2, int pos, int len);

sub-string

- substr(int pos, int len);

alter string

- assign();
- insert(const string&, int len);
- insert(int pos, const string& s);
- erase();
- append();
- replace();

search string

- find()

Pointers to Objects

Pointers to Objects

- string s = "hello";

- string* ps = &s;

Operators with Pointers

- &: get address
  - ps = &s;
- *: get the object
  - (*ps).length()
- ->: call the function
  - ps->length()

Two Ways to Access

- string s;
  - s is the object itself
- string *ps;
  - ps is a pointer to an object

Object vs Pointer

- string s;
  - At this line, object s is created and initialized
- string *ps;
  - At this line, the object ps points to is not known yet.

Assignment

- string s1, s2;
  - s1 = s2;
- string *ps1, *ps2;
  - ps1 = ps2;

dynamically allocated memory

Dynamic memory allocation

- **new**
  - new int;
  - new Stash;
  - new int[10]
- **delete**
  - `delete p;`
  - `delete[] p;`

new and delete

- **new** is the way to allocate memory as a program runs. Pointers become the only access to that memory

- **delete** enables you to return memory to the memory pool when you are finished with it.

Dynamic Arrays

```
int * psome = new int [10];
```

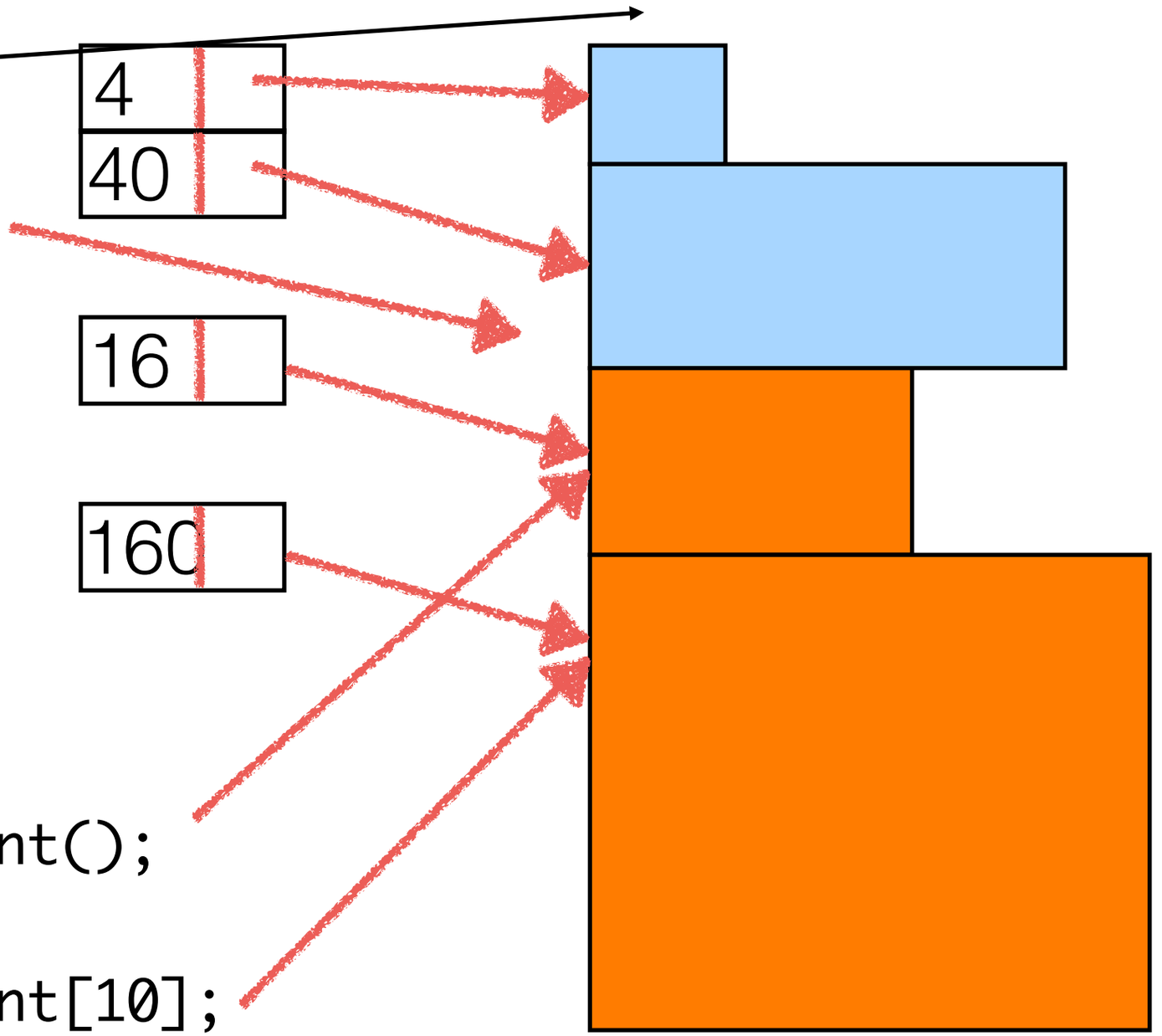
- The **new** operator returns the address of the first element of the block.

```
delete [] psome;
```

- The presence of the brackets tells the program that it should free the whole array, not just the element

The new-delete mech.

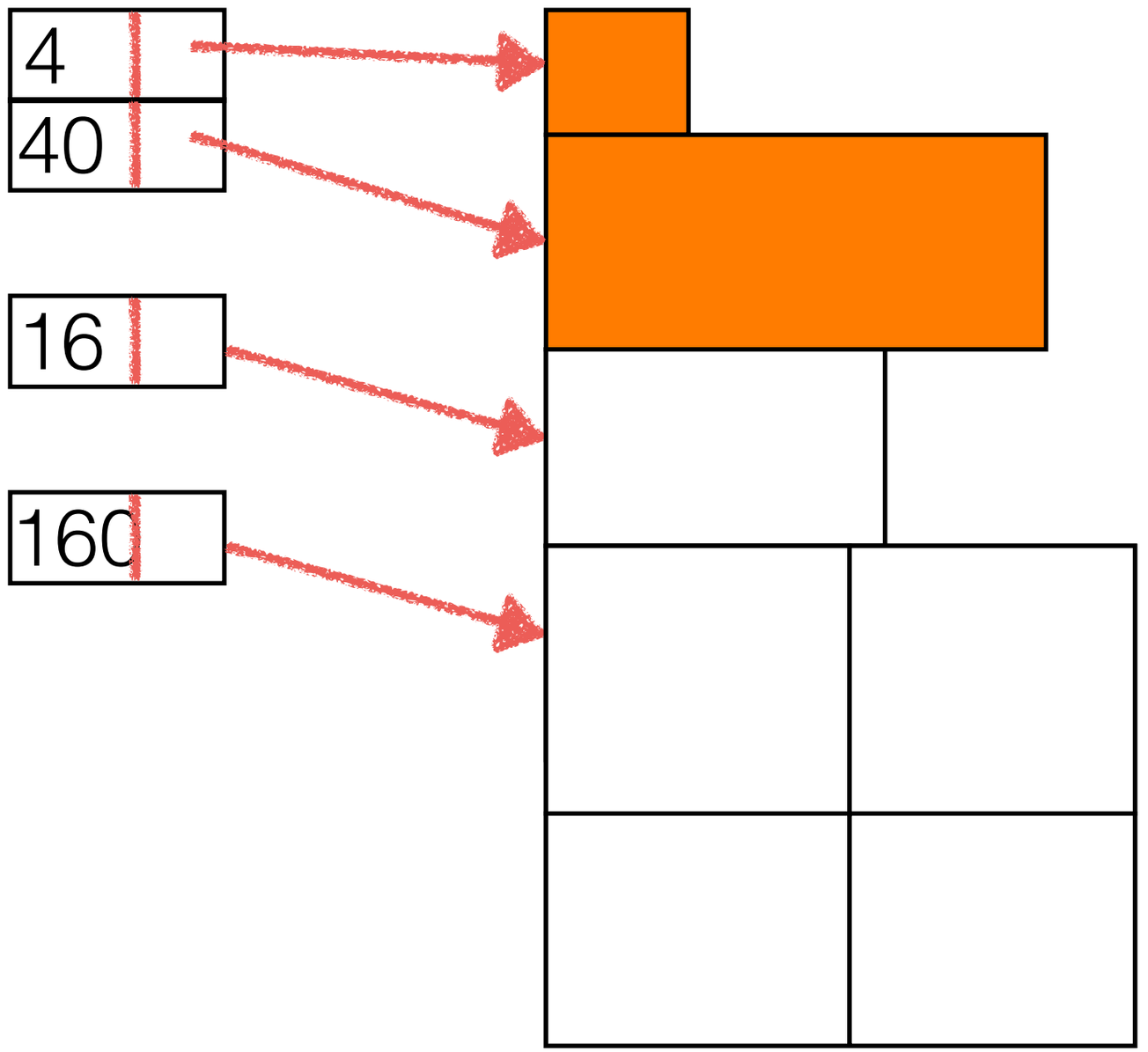
```
int *p=new int;

int *a=new int[10];
```




```
Student *q=new Student();

Student *r=new Student[10];
```

The new-delete mech.

```
int *p=new int;

int *a=new int[10];

Student *q=new Student();

Student *r=new Student[10];

delete p;

a++;delete[] a;

delete q;

delete r;

delete[] r;
```



Tips for new and delete

- Don't use delete to free memory that new didn't allocate.
- Don't use delete to free the same block of memory twice in succession.
- Use delete [] if you used new [] to allocate an array.
- Use delete (no brackets) if you used new to allocate a single entity.
- It's safe to apply delete to the null pointer (nothing happens).