USTC

Chapter 4
Computation

王子磊 (Zilei Wang)
Email: zlwang@ustc.edu.cn
http://vim.ustc.edu.cn

Overview

- 本章介绍一些计算相关的基本概念
  - 核心是让大家能够编写出正确而且规范的程序
  - 什么是计算？怎么样计算最好？
- 编程的构造和思想(简单有用)
  - 程序的序列执行顺序 Sequential Execution
  - 表达式和语句 Expressions and Statements
  - 选择 Selection
  - 迭代 Iteration
  - 函数 Functions
  - 向量 Vectors

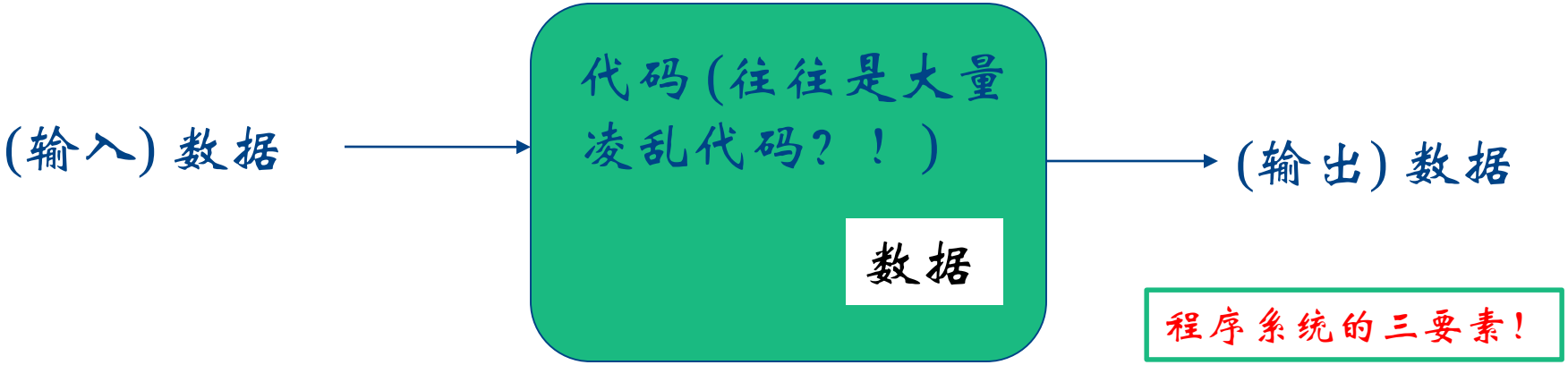
这些概念你已经知道了！

- Note:
  - 算术运算
    - d = a+b*c
  - 判断选择

    “if this is true, do that; otherwise do something else ”
  - 如何迭代(循环)
    - “do this until you are finished”
    - “do that 100 times”
  - 使用函数功能
    - “go ask Joe and bring back the answer”
    - “hey Joe, calculate this for me and send me the answer”

- 我们现在要讲的就是这些你已经知道的概念和相应的语法(深入理解)

Computation



- 输入 Input
  - 从键盘、文件、其它输入设备、其它程序（进程间通信）、程序的其它部分（线程间通信或函数调用）
- 计算 Computation
  - 程序从输入到产生输出过程中具体的操作
- 输出 Output:
  - 到屏幕、文件、其它输出设备、其它程序、程序的其它部分

Computation

- 程序员的任务是将计算表示出来
  - 正确 Correctly ——包括正常和异常流程
  - 简单 Simply
  - 高效 Efficiently

容易忽略之处！

- 一种手段是分治
  - 把一个大问题分为几个小问题分别解决
  - 直到问题小到能够被我们很好地理解和解决为止 (原则)
- 另一种手段是抽象
  - 提供一种高级的概念，而将具体实现细节隐藏在接口之后
- 数据的良好组织是构成好代码的关键
  - 输入/输出格式 Input/output formats
  - 协议 Protocols
  - 数据结构 Data structures
- 强调：结构和组织非常重要
  - 简单语句的堆砌并不能构成好的代码

语言特性

- 每一种编程语言特性都表示了一种思想
  - For example
    - + ：加
    - * ：乘
    - **if (*expression*) *statement* else *statement* ;** 选择
    - **while (*expression*) *statement* ;** 迭代
    - **f(x);** 函数
    - …
- 将许多种语言特性结合起来，就能写出有用的程序
  - 理论上可以完成任何任务

表达式

```
// compute area:
int length = 20;          // 最简单的表达式: 字面常量(here, 20)
                          // (here used to initialize a variable)
int width = 40;
int area = length*width;          // 乘法
int average = (length+width)/2;   // 加法和除法
```

- 优先级应用的规则

  如：a*b+c/d 代表(a*b)+(c/d)，而不是 a*(b+c)/d.

- 如果对运算符优先级不确定，就用括号，但不要滥用(原则)
- 不要使用非常复杂的表达式

  ```
  a*b+c/d*(e-f/g)/h+7          // too complicated
  ```

- 再次说明：变量选择有意义的名字

表达式

- 表达式由操作符和操作数构成
  - 操作符指定进行何种操作
  - 操作数指定操作的数据对象(一种类型)
- 布尔类型: **bool** (**true** and **false**)
  - 等于: ==(等于), !=(不等于)
  - 逻辑: &&(与), ||(或), !(非)
  - 关系: <(小于), >(大于), <=, >=
- 字符类型: **char** (e.g., '**a**', '**7**', and '**@**')
- 整数类型: **short, int, long**
  - 算术运算: +, -, *, /, % (余)
- 浮点类型: e.g., **float, double** (e.g., **12.45** and **1.234e3**)
  - 算术运算: +, -, *, /

简约运算符

- 多数二元运算符有对应等价的简约运算符
  - 例如：
    - **a += c** 表示 **a = a+c**
    - **a *= scale** 表示 **a = a*scale**
    - **++a** 表示 **a += 1**
      or **a = a+1**
  - 简约运算符 (“Concise operators”) 通常更方便
    - 更清晰直观，可读性强
    - 这种编程方式直接体现程序思想 (功能效果而非实现)

语句

- 一条语句
  - 是以分号“;”结束的表达式，或是
  - 一个声明，或是
  - 一条控制语句，控制程序流

#include “file.h”是语句吗？

- 举例：
  - **a = b;**
  - **double d2 = 2.5;**
  - **if (x == 2) y = 4;**
  - **while (cin >> number) numbers.push_back(number);**
  - **int average = (length+width)/2;**
  - **return x;**

后续你将更加理解这些语句的含义……

选择语句

- 有时我们需要在多个情况下进行选择 (Selection)
- 比如，从两个值中选择一个较大的
  - 可以通过if语句来实现

常见错误，但编译器不会报错！

```
if (a<b)              // Note:  No semicolon here
  max = b;
else                  // Note:  No semicolon here
  max = a;
```

- 语法规则是

```
if (condition)
   statement-1      // 如果条件满足，则执行 statement-1
else
   statement-2      //如果条件不满足，则执行 statement-2
```

循环语句 (while 循环)

- 世界上第一台能存储程序的计算机 (EDSAC) 上运行的第一个程序就是循环语句 (David Wheeler, Cambridge, May 6, 1949)

```
// calculate and print a table of squares 0-99:
int main()
{
    int i = 0;
    while (i<100) {
      cout << i << '\t' << square(i) << '\n';
      ++i ;      // increment i
    }
}
// (No, it wasn't actually written in C++ ☺.)
```

循环语句 (while 循环)

- While语句需要
  - 一个循环变量(控制变量); here: **i**
  - 初始化控制变量; here: **int i = 0**
  - 一个循环终止条件; here: if **i<100** is false, terminate
  - 递增控制变量; here: **++i**
  - 循环中完成的操作（循环体）; here: **cout << …**

```
int i = 0;
while (i<100)  {
        cout << i << '\t' << square(i) << '\n';
        ++i ;      // increment i
}
```

应该一直保持的习惯！

循环语句 (for 循环)

- 另一种循环语句形式: **for** 循环
- 可以将所有的控制信息放置在一起 (**for**语句)
  - 更容易理解和维护

```
for (int i = 0; i<100; ++i) {
    cout << i << '\t' << square(i) << '\n';
}
```

Note: what is **square(i)**?

语法结构为：

for (initialize**;** condition **;** increment **)**
controlled statement

- 不要在for语句的循环体内修改循环控制变量的值 (合法但不合理)!

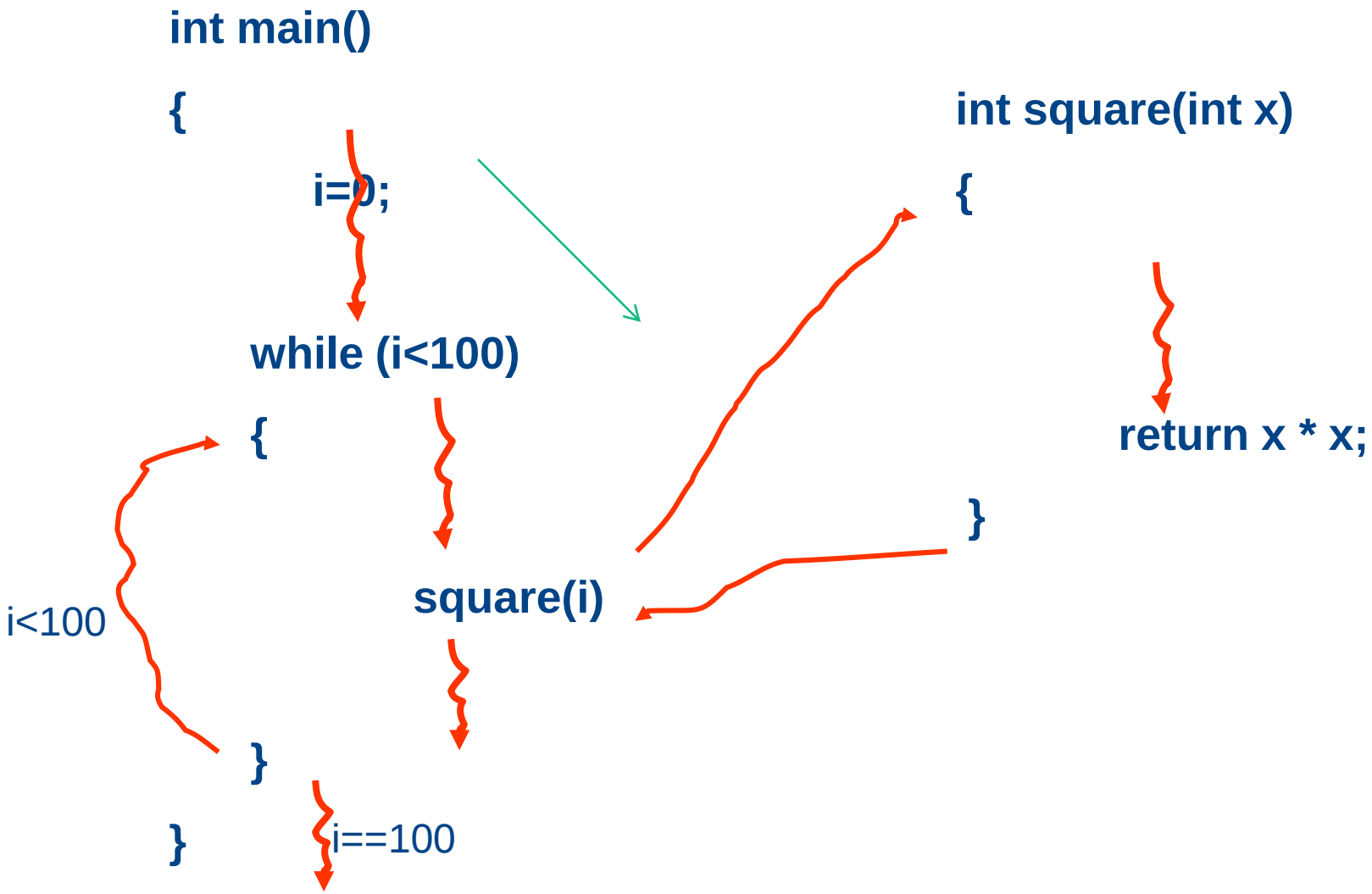
函数

- 上面程序中，square(i)是什么？
  - square()是一个函数调用

```
int square(int x)
{
    return x*x;
}
```

- 当我们将一部分计算任务独立实现的时候，可以定义一个函数，这样可以(什么时候使用函数的原则)：
  - 实现计算逻辑的分离 → 抽象，隐藏实现细节
  - 使代码更清晰(通过使用函数名)
  - 使得同样的代码在程序中可以多次使用
  - 减少程序调试的工作量

控制流


int main()
{
i=0;
while (i<100)
{
i<100
square(i)
}
}
i==100
int square(int x)
{
return x * x;
}

函数

- 前面的函数定义：

```
int square(int x)
{
   return x*x;
}
```

- 函数定义 (function definition) 的语法：

```
Return_type  function_name ( Parameter list )
                                          // (type name, etc.)
{
   // use each parameter in code
   return some_value;                     // of Return_type
}
```

函数四要素

Another Example

- 前面例子中是找出两个值中的最大值，现在我们定义为一个函数形式，并返回最大值：

```
int max(int a, int b)          // this function takes 2 parameters
{
    if (a<b)
        return b;
    else
        return a;
}

int x = max(7, 9);             // x becomes 9
int y = max(19, -27);          // y becomes 19
int z = max(20, 20);           // z becomes 20
```

迭代数据结构- Vector

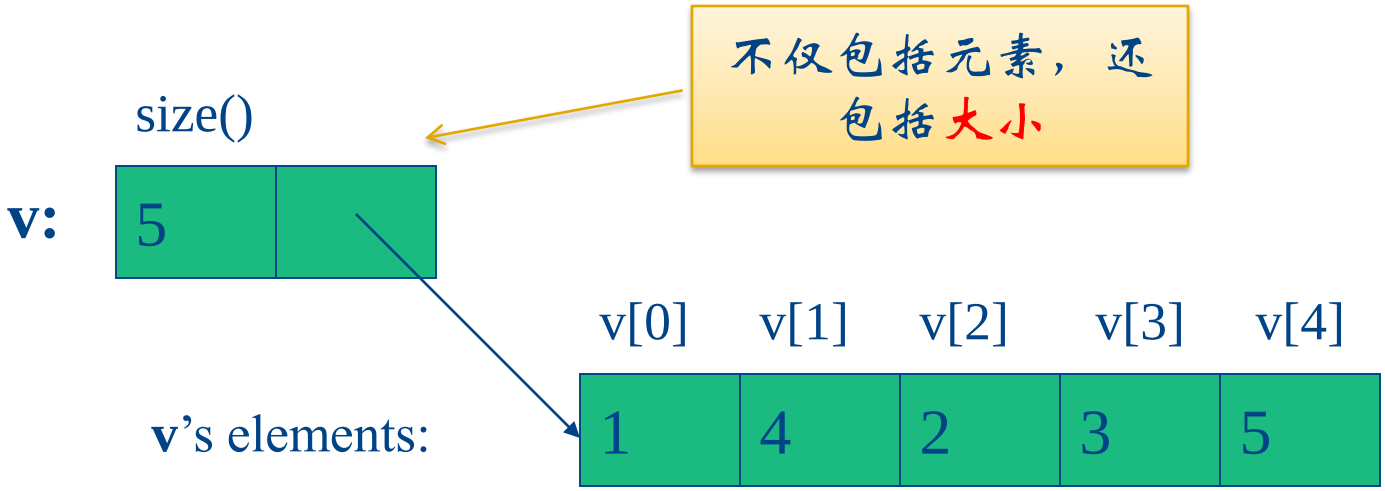
- 在编程中，我们通常需要处理一组数据
  - 能够用**vector**去存储这些数据，如：

```
// read some temperatures into a vector:
int main()
{
    vector<double> temps;          // declare a vector of type double to store
                                   // temperatures – like 62.4
    double temp;                   // a variable for a single temperature value
    while (cin>>temp)              // cin reads a value and stores it in temp
        temps.push_back(temp);     // store the value of temp in the vector
    // ... do something ...
}
```

- 如果正确输入数据，**cin>>temp** 会返回**true**，否则返回**false**，直到达到文件结尾或终止输入

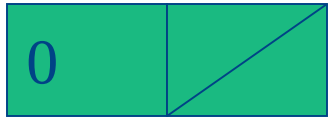
向量 Vector

- Vector 是标准库中最常用的数据类型
  - **vector<T>** 是一个T类型值的序列
  - 一个向量只能存储与其数据类型相同的数据
  - 可以想象一个名字为v的**vector**，包含五个元素: {1, 4, 2, 3, 5}:

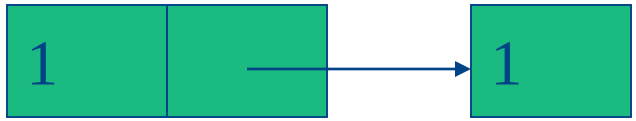


向量 Vector

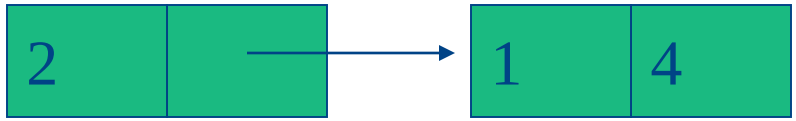
vector<int> v; // start off empty

v: 

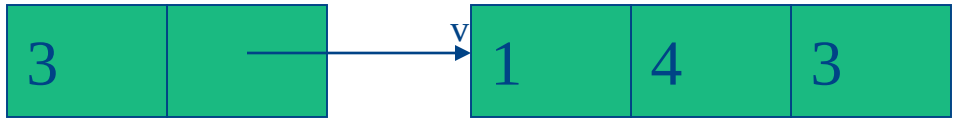
v.push_back(1); // add an element with the value **1**

v: 

v.push_back(4); // add an element with the value **4** at end ("the back")

v: 

v.push_back(3); // add an element with the value **3** at end ("the back")

v: 

向量 Vector

- vector中存储的数据很容易进行操作:

```
// compute mean (average) and median temperatures:
int main()
{
    vector<double> temps;          // temperatures in Fahrenheit, e.g. 64.6
    double temp;
    while (cin>>temp)  temps.push_back(temp); // read and put into vector

    double sum = 0;
    for (int i = 0; i< temps.size(); ++i) sum += temps[i];
                                                  // sums temperatures
    cout << "Mean temperature: " << sum/temps.size() << endl;

    sort(temps.begin(), temps.end());
    cout << "Median temperature: " << temps[temps.size()/2] << endl;
}
```

组合语言特性

- 通过组合语言特性、内建类型、用户自定类型等，你可以写出许多新的有意思的程序
  - 目前，我们已经学习了
    - **bool, char, int, double类型的变量和字面常量**
    - **vector, push_back(), [ ] (下标)**
    - **!=, ==, =, +, -, +=, <, &&, ||, !**
    - **max( ), sort( ), cin>>, cout<<**
    - **if, for, while**
  - 下面，我们使用这些语言特性，写一些不一样的程序
    - Let's try to use them in a slightly different way…

Example – Word List

```
// "boilerplate" left out

vector<string> words;
string s;
while (cin>>s && s != "quit")          // && means AND
        words.push_back(s);

sort(words.begin(), words.end());      // sort the words we read

for (int i=0; i<words.size(); ++i)
        cout<<words[i]<< "\n";

/*
    read a bunch of strings into a vector of strings, sort
    them into lexicographical order (alphabetical order),
    and print the strings from the vector to see what we have.
*/
```

Word list – 取消重复

```
// Note that duplicate words were printed multiple times.  For
// example "the the the".  That's tedious, let's eliminate duplicates:

   vector<string> words;
     string s;
     while (cin>>s && s!= "quit") words.push_back(s);

     sort(words.begin(), words.end());

     for (int i=1; i<words.size(); ++i)
           if(words[i-1]==words[i])
                 "get rid of words[i]"     // (pseudocode)
     for (int i=0; i<words.size(); ++i) cout<<words[i]<< "\n";

//   there are many ways to "get rid of words[i]"; many of them are messy
//   (that's typical).  Our job as programmers is to choose a simple clean
//   solution – given constraints – time, run-time, memory.
```

Word list - 取消重复!

```
// Eliminate the duplicate words by copying only unique words:
    vector<string> words;
    string s;
    while (cin>>s && s!= "quit")   words.push_back(s);
    sort(words.begin(), words.end());
    vector<string>w2;
    if (0<words.size()) {                          // Note style { }
          w2.push_back(words[0]);
          for (int i=1; i<words.size(); ++i)
                if(words[i-1]!=words[i])
                     w2.push_back(words[i]);
     }
    cout<< "found " << words.size()-w2.size() <<  " duplicates\n";
    for (int i=0; i<w2.size(); ++i)  cout << w2[i] << "\n";
```

算法

- 我们只是用了一个简单的算法
- 算法定义 (from Google search)
  - "a logical arithmetical or computational procedure that, if correctly applied, ensures the solution of a problem." – *Harper Collins*
  - "a set of rules for solving a problem in a finite number of steps, as for finding the greatest common divisor." – *Random House*
  - "a detailed sequence of actions to perform or accomplish some task. Named after an Iranian mathematician, Al-Khawarizmi. Technically, an algorithm must reach a result after a finite number of steps, …The term is also used loosely for any sequence of actions (which may or may not terminate)." – *Webster's*
- 在前例中，我们通过**sort vector**来取消重复
  - 重复值会相邻
  - 只copy与前一个值不同的元素

Next

- 如何处理编程中的错误