第1章 C++入门

目录

- 1.1 从C到C++
- 1.2 程序与语言
- 1.3 结构化程序设计
- 1.4 面向对象程序设计
- 1.5 程序开发过程
- 1.6 最简单的程序
- 1.7 函数

学习目标

- 了解C++从C演变而来的历史
- 了解不同类型的程序设计语言
- 了解编程中的几个步骤
- 理解在编程开发环境中的各项要素
- 了解程序中的函数概念

1.1 从C到C++

- **计算机**
  - 计算机是能以人几百万甚至几十亿倍速度进行计算并作出逻辑判断的设备。
- **计算机程序**
  - 计算机在一组指令控制下处理数据，这组指令称为计算机程序。
- **硬件**
  - 计算机由各种设备组成
    - 键盘，屏幕，鼠标，磁盘，内存，光驱，处理器，…
    - 我们把这些设备称为硬件。
- **软件**
  - 计算机上运行的计算机程序被称为软件

C语言的历史

- C语言
  - 由两种早期语言BCPL 和 B发展而来的。
  - 作为 UNIX操作系统的开发语言，并用来开发现代操作系统。
  - 具有硬件无关性
    - 可以方便的移植到大多数计算机上
  - 20世纪70年代后期，C语言发展成为 我们现在所说的“传统C语言”
- 标准化
  - 1983年，正式定名C++
  - 1989:推出ANSI 标准
  - 1999: 标准进行更新
    - ANSI/ISO 9899: 1990

1.2 程序与语言

- 三种计算机语言
  - 1. **机器语言**
    - 计算机能够直接识别的语言
    - 是特定计算机的自然语言
    - 由计算机的硬件设计定义
    - 通常由一系列数字组成
      - 最终简化为0和1
    - 让计算机执行最基本的操作
      - 一次一个

例如:

+1300042774
+1400593419
+1200274027

加班工资和基本工资相加的一段程序，然后把结果存入工资总额

•对程序员而言太繁琐

- 2. 汇编语言
  - 类似英文缩写的助记符来表示计算机的基本操作
  - 对程序员比较清晰
  - 计算机不能直接理解
    - 由汇编器转换为机器语言
  - 实例:

```
mov eax, DWORD PTR a_$[eb
add ecx,ecx
Mov DWORD PTR a_ $[ebp]
```

缺点:

汇编语言进行程序设计仍然需要很多指令才能够实现最简单的任务。

3. 高级语言

- 高级语言类似日常英语，包含有常用的数学符号，一条语句完成大量任务。
- 实例:
- `a = 3*a-2*b+1;`

把高级语言转化为机器语言的翻译程序称为编译器 (compilers)

1.3 结构化程序设计

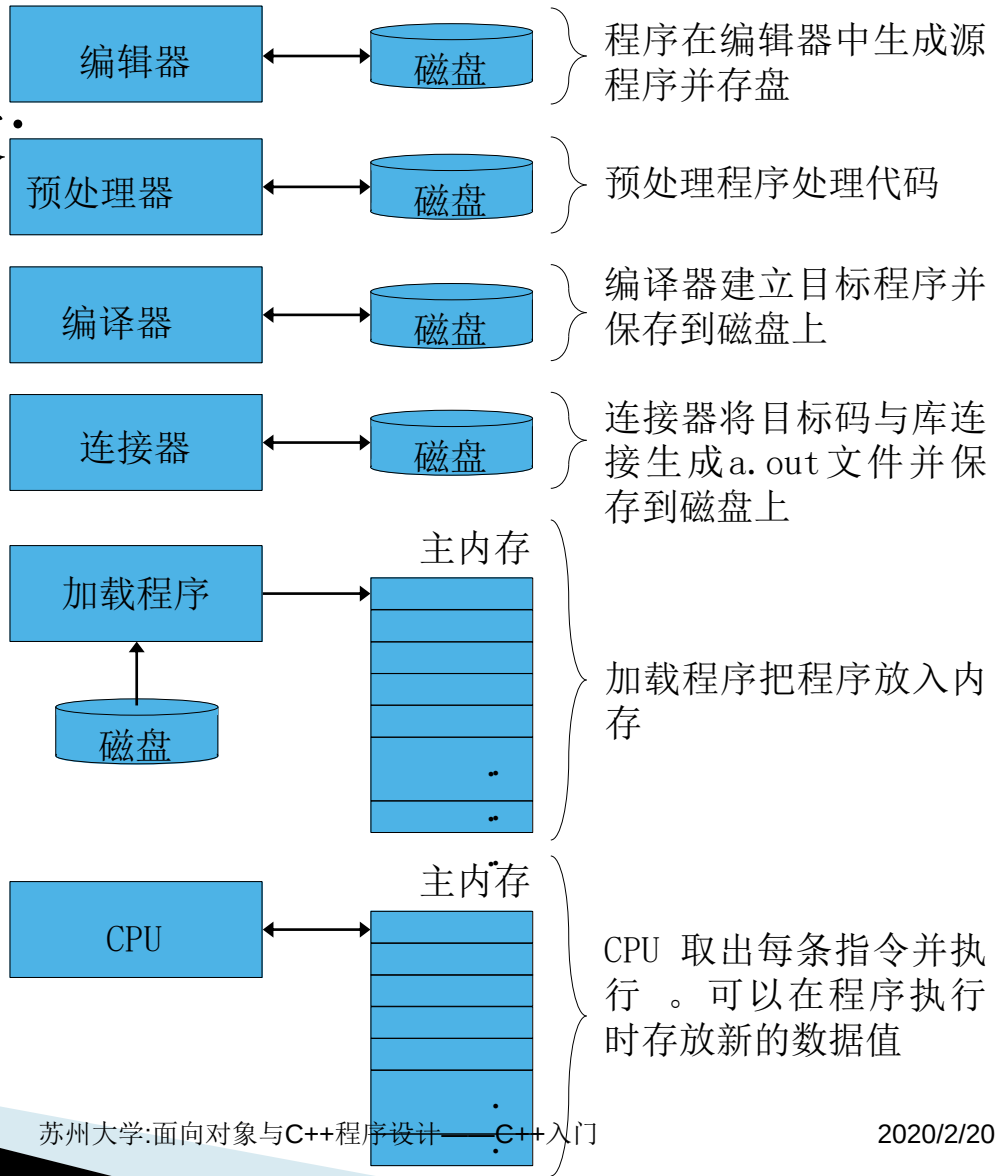
- 功能分解并逐步求精
- 程序由名为函数的模块或片段所组成的。
  - 程序员可以开发自己的函数
    - 优点：可以确切地知道这些函数如何工作；
    - 缺点：开发新函数耗费大量的时间。
  - 避免了一切重头开始
    - 如果有库函数存在，最好使用库函数
    - 标准库函数都是经过仔细编写的，能够有效地执行

1.4 面向对象程序设计

- 封装和数据隐藏
- 继承和重用
- 多态性

1.5 程序开发过程

- C++语言的6个阶段：
  1. *编辑*
  2. *预处理*
  3. *编译*
  4. *链接*
  5. *加载*
  6. *执行*



1.6 最简单的程序

- 显示一行文本

以//开始，表示注释。

```
1   // ch1_1.cpp  第一个程序
3   #include<iostream>
4   using namespace std;
5   // 程序从函数main开始执行
6   int main()
7   {
8       cout<<"I am a student.\n";
9   } // 函数main结束
```

目的为了增加程序可读性，计算机不会执行注释中的语句

```
I am a student.
```

- **#include<iostream>**
  - 这是一个C++预处理指令。
  - 这一行告诉预处理器把标准输入/输出头文件（iostream）包括到这个程序中。
  - 头文件中包含了在编译诸如cout与cin的信息和声明。
- **注释**
  - //： 单行注释方式，注释符号后面的内容都是注释
    - `// a = b+c;`
    - `a =fun(5);  // 计算5的阶层`
  - /*注释内容*/：注释一段代码，可以是一行或多行，也可以是一行中的一部分。
    - `/* a = b+c;`
    - `c = a*2; */`
    - `a = b+c;  /*计算b+c的结果*/`

补充

- **#include<iostream.h>**
- 这是老旧C++表述，C++98标准之后，C++标准头文件均去掉.h，并在全部包含语句结束后，添加using namespace std; 语句，即：

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

补充

- C++继承了C的诸多资源，资源通过包含头文件来使用。所以C++中有些头文件来自C，例如：math.h。C++对其进行整合，整合的方式是在头文件之前冠以c,再去掉.h，例如：#include<iostream>

```
//C++头文件
#include<cmath>          //C头文件整合
using namespace std;
```

- C++兼容C，若采用C头文件，则其使用按C语言语法规则，两者存些微差别。

- #include<iostream>
  using namespace std;　　//因含C++头文件而需表述
  #include<math.h>　　// C头文件

- int main()
  - C/C++程序包含有多个函数，但是仅有且必须有一个main函数。
  - main函数是任何C和C++程序的入口点
  - 圆括号表示这是一个函数。
  - int 意味着 main 函数 “返回一个整型的值。
  - 花括号{...}表示块

- return 0;
  - 关键词return是我们用来表示退出程序的几种方法之一。
  - return 0 表示程序成功结束（在C++中，main函数结束时的return 0;语句可以省略）。
- 右大括号 }
  - 表示到达了main的结尾。
- 形如int main(){…}为一个函数的完整描述

1.7 函数

- C与C++在函数结构上已经趋向一致
- 形如type funcname(){…}为一个函数的形式描述，其中

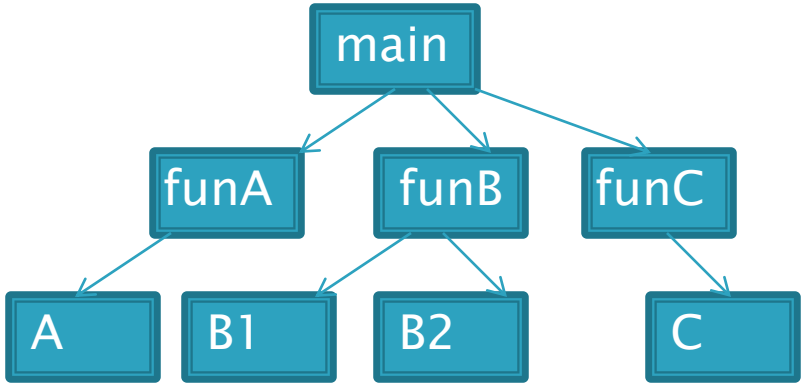
  type为数据类型，例如int，

  funcname为函数名，例如main

  ()中描述传递的数据参数，后面慢慢展开

  {}中描述需要执行的命令（动作序列）

- C++用函数组织程序
  函数规定动作的执行次序
- C++程序是函数驱动的
  可以在程序中
  定义一堆函数，
  从main函数始，
  调用其他函数



- 命令（动作）描述中可以对函数直接调用,例如：
  c=max(a,b);
- funcname(...)的形式称为函数调用
- 被调用的函数必须在调用之前有声明,例如：
  ```
  double max(double x, double y);
  ```
- type funcname(){...}称为函数定义,例如：
  ```
  double max(double a, double b)
  {
    if(a>b) return a;
    else return b;
  }
  ```
- 函数声明就是取函数定义去掉花括号的部分,再添上分号

```
//**ch1_3.cpp**
#include<iostream>
#include<cmath>
using namespace std;
double max(double x, double y);

int main(){
  double a,b,c;
  cout<<"input two numbers:\n";
  cin>>a>>b;

  c=max(a,b);

  cout<<"the squart of maximum="<<sqrt(c);
}

double max(double x, double y){
  if(x>y) return x;
  else    return y;
}
```

返回类型Type 描述为double

函数调用前必须要有函数声明

函数调用时括号中实际传递的值称为实际参数

调用math函数库中的sqrt函数,其声明用：#include<cmath>

函数声明或定义的括号中的参数称为形式参数

多选题 1分

设置

以下表述哪些是正确的( )

A 一个C++程序只能有一个函数

B 一个C++程序只能有一个main函数

C 一个C++程序可以有多个函数，包括多个main函数

D 一个C++程序可以有多个函数，但只能有一个main函数。

提交

多选题 1分

设置

下面描述正确的是( )

A //可以用于注释一行的中间内容

B /* */可以用于注释一行的中间内容

C C++程序中注释和代码一样会被编译器编译

D C++程序中注释不会被编译器编译

提交