

感性认识C/C++及程序设计

Wang Houfeng

wanghf@pku.edu.cn

EECS, PKU

内容

➤ 程序设计的基本思想

- 程序语言的受控特点
- 如何学好编程

认识一个C++程序

```

/*****
/*  example.cpp  */
*****/
#include<iostream>

using namespace std;

int main( )
{
    int number[45] = {78, 56, 69, 31, 36, 67, 31, 47, 69, 34, 45, 74, 61, 82, 43, 41, 76, 79,
                      81, 66, 54, 50, 76, 51, 53, 28, 74, 39, 45, 61, 52, 41, 43, 75, 78, 84, 72, 51, 43, 64, 75,
                      81, 69, 55, 74};
    int max = 0;
    int i = 0;
    for(i = 0; i < 45; i++)
    {
        if(number[i] > max)
            max = number[i];
    }
    cout<<"The Maximal Number is:"<<max;
    return 0; //函数结束返回
}

```

注释

预编译：文件包含命令 (`<>`, “ ”)

声明名字空间 (避免变量、函数等重名冲突)

函数名

变量定义

循环控制语句

条件（分支）控制语句

赋值语句

输入输出语句

函数体

程序设计

- 用一种特定的语言（如：C/C++）编写程序
- 如何才能设计好程序
 - 掌握编程语言（如C/C++）**【语言学家】**
 - 运用编程语言解决实际问题**【文学家】**
 - 长期训练，提升技能**【多写多练】**

理解程序设计

- 程序设计的本质

- 如何让计算机对数据自动加工（运算），以解决用户期望的任务

- 程序设计最直接的两方面问题

- 数据操作

- 数据运算
- 数据存储
- 数据传输

操作性语句
(可执行语句)

- 数据表示

- 输入数据表示
- 输出数据表示
- 运算中的临时表示

非操作性语句
(非可执行语句)

程序设计语言的构成

- 4种基本成分 vs. 4大基本构建

可执行语句

- 数据成分，用以描述程序中所涉及的数据【存储】
- 运算成分，用以描述程序中所包含的运算【运算】
- 控制成分，用以表达程序中的控制构造【控制】
- 传输成分，用以表达程序中数据的传输【传输】

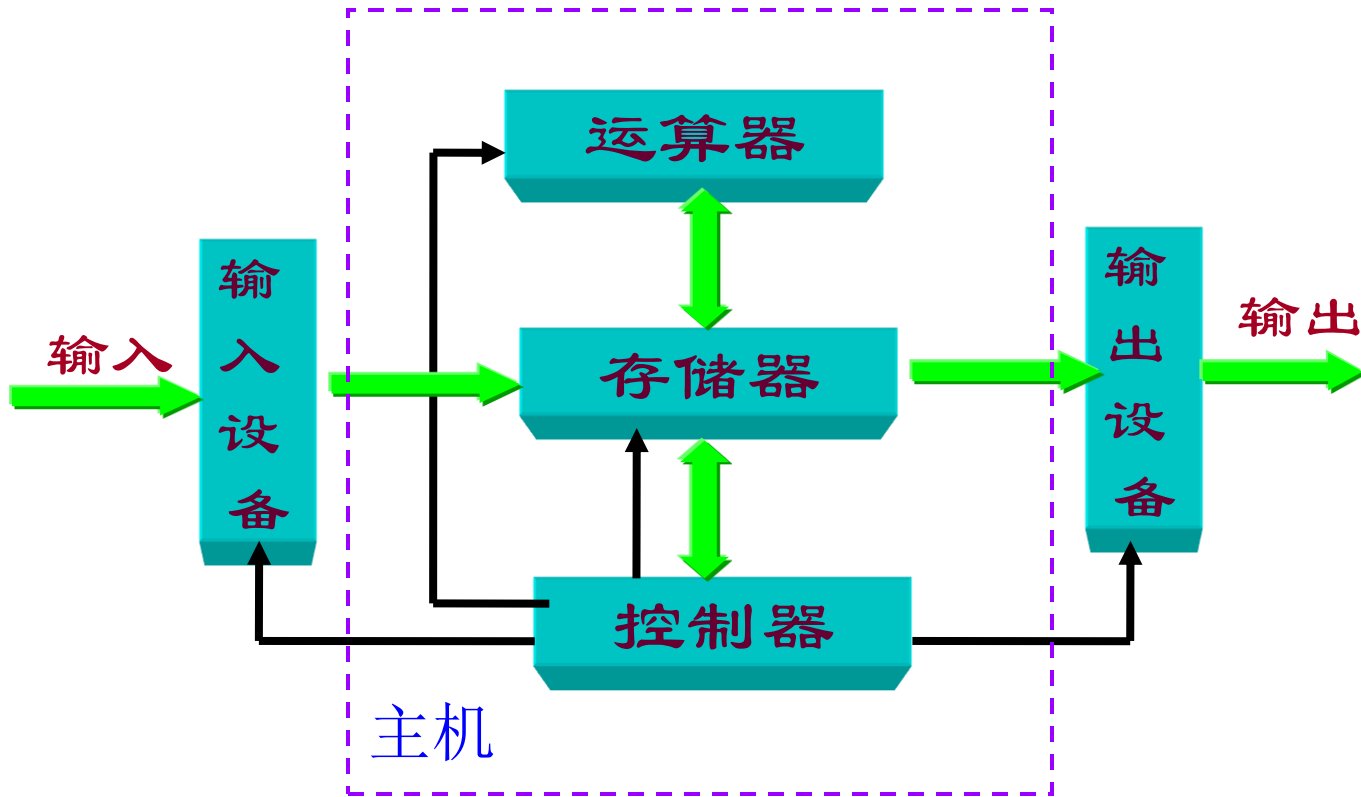
计算机的基本构成

四大部件

——计算机科学技术百科全书

通过上述4种成分描述人们在现实世界中的不同需求

冯.诺依曼体系结构



程序设计中的四种成分与作用

- 数据表示

- 输入数据的表示
- 输出数据的表示
- 中间数据的表示

数据成分

- 数据操作：操作与控制

- 数据运算
- 数据传输
- 数据加工流程控制

运算成分

输入输出成分

控制成分



程序设计的核心： 数据加工

- 数据结构设计

- 如何表示纷繁复杂的数据

- 输入数据表示
 - 输出数据表示
 - 中间运算过程中的数据表示

非操作性语句
(非可执行语句)

- 算法设计

- 如何加工数据

- 数据的运算
 - 对数据运算的控制

操作性语句
(可执行语句)

看一个例子

◆ 给你一个数列：

78, 56, 69, 31, 36, 67, 31, 47, 69, 34, 45, 74, 61,
82, 43, 41, 76, 79, 81, 66, 54, 50, 76, 51, 53, 28,
74, 39, 45, 61, 52, 41, 43, 75, 78, 84, 72, 51, 43,
64, 75, 81, 69, 55, 74

◆ 你如何找出其中最大的数字

我们可以这样操作

- 基本思想
 - ✓ 从左到右逐一比较，选择最大的数
- 一种求解步骤
 - ✓ 逐次取一个数，与当前为止的最大数比较，如比当前最大数还要大，当前数变是目前所取的数，否则不变
 - ✓ 重复上述过程直到把所有的数都取出并比较完毕
 - ✓ 最终保留下来的最大数即为所求的数

再稍微整理一下

程序设计：将下面思想用程序语言表达出来

- 重新描述
 - 在大脑中开辟一片“存储空间”存放所有数
 - 使用一个“存储空间”存放“到此为止的最大数”
 - 从存储空间逐次取数，直到最后一个，重复以下操作
 - 比较“存储空间中的数字”与“当前最大数”
 - 如果“存储空间中的数字”大于“当前最大数”
 - 那么，将“当前最大数”换成“存储空间中的数字”
 - 输出“当前最大数”

下面的程序可以实现吗

```
#include<iostream>
using namespace std;
int main( )
{
```

设想有个集合表示（存储）所有数

```
int number[45] = {78, 56, 69, 31, 36, 67, 31, 47, 69, 34, 45, 74, 61,
82, 43, 41, 76, 79, 81, 66, 54, 50, 76, 51, 53, 28, 74, 39, 45,
61, 52, 41, 43, 75, 78, 84, 72, 51, 43, 64, 75, 81, 69, 55, 74};
```

```
int max = 0;
```

用于记忆当前为止的最大值，
为什么初始值为0

```
int i = 0;
```

```
for(i = 0; i < 45; i++)
```

```
{
```

```
    if(number[i] > max)
```

```
        max = number[i];
```

逐一比较（需要控制比较次数）

```
}
```

```
cout<<"The Maximal Number is:"<<max;
```

```
return 0;
```

```
}
```

内容

- 程序设计的基本思想
- 程序语言的受控特点
- 如何学好编程

计算机如何理解人的意图

- 按照计算机可理解的方式表达数据处理过程
 - 一般的中英文表达方式计算机难以理解
 - 自然语言有太多的歧义：能穿多少就穿多少
 - 自然语言有太多隐藏的知识：蓝色的海洋 vs. 红色的海洋
- 程序设计语言：计算机可以理解的语言
 - 自然语言的子集
 - 有严格含义（语义）的受控语言
 - 单词受控：系统规定的单词和用户引入的单词均受控
 - 数据与运算受控
 - 语句表达形式受控
 - 整体结构受控

计算机如何理解人的意图

- 用计算机可理解的语言写程序
- 程序设计满足语言受控、语义简明的基本要求
 - **单词受控**：系统给定的单词和用户按规定引入的单词
 - **数据与运算受控**：数据类型和运算类型有严格规定
 - **语句表达形式受控**：语句表达形式有严格规定
 - **逻辑结构受控**：3种基本结构+程序构成规定

语言成分的构成对应于计算机功能部件的构成

C/C++的单词受控（1）

- 系统特定的单词受控（保留字/关键字）

—每个单词有**特定的唯一含义**，**没有歧义**

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	unsigned	union	void
volatile	while	bool	catch	class

C/C++的关键字（亦称为保留字，即系统保留使用）

关键词的三种主要功能

牢记这些单词的功能

auto 不可执行	break 控制作用 可执行	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	unsigned	union	void
volatile	while	bool	catch	class

定义型：定义数据类型与数据变化约束

控制型：控制数据加工的流程

特殊运算：sizeof

用户引入单词受控

- 程序员引入的单词受控 (引入4种单词)
 - 存放数据的变量和常量
 - 引入新的类型
 - 引入函数 (独立的加工子模块)
 - 引入类 (数据与函数的集成)
- 上述4类单词的引入均由**定义类关键词**限定
 - 引入整型变量: **int** a;
 - 引入新的数据类型: **struct usertype**{ int a; int b;}
 - ...
- 程序员引入的单词意义明确 (由**关键词控制**)

新单词的含义受控

标识符：用户引入新单词

- 定义：用来标识变量、常量、函数等的名字
- 标识符的构成：
 - 只能由字母、数字、下划线组成，且第一个符号必须是字母或下划线
 - 大小写是有区别的
 - 长度没有限制，尽量通过字面表示含义
 - 不能使用关键字（关键字为系统保留使用）

新单词的表示形式受控

例：判断下列标识符号合法性

sum Sum M.D.John day Date 3days
student_name #33 lotus_1_2_3
char a>b _above \$123

常量 & 变量标识符

- 数的两种形式：
 - 常量：值不变的量
 - 变量：值可以变化的量，必须用标识符（变量名）表示
- 变量在使用前，必须**先声明**，一般格式为：
数据类型 变量1[, 变量2, ..., 变量n];

```
#include <iostream>
using namespace std;
int main()
```

**标识符有限定的含义
让计算机实现限定操作**

```
{  const int num=3;
    float total;
    char ch1,ch2='D';
    total=num*15;
    ch1=ch2-'A'+'a';
    cout<<"total="<<total<<"ch1="<<ch1;
    return 0;
}
```

← **标识符表示常量**

} **先声明变量，可以赋初值**

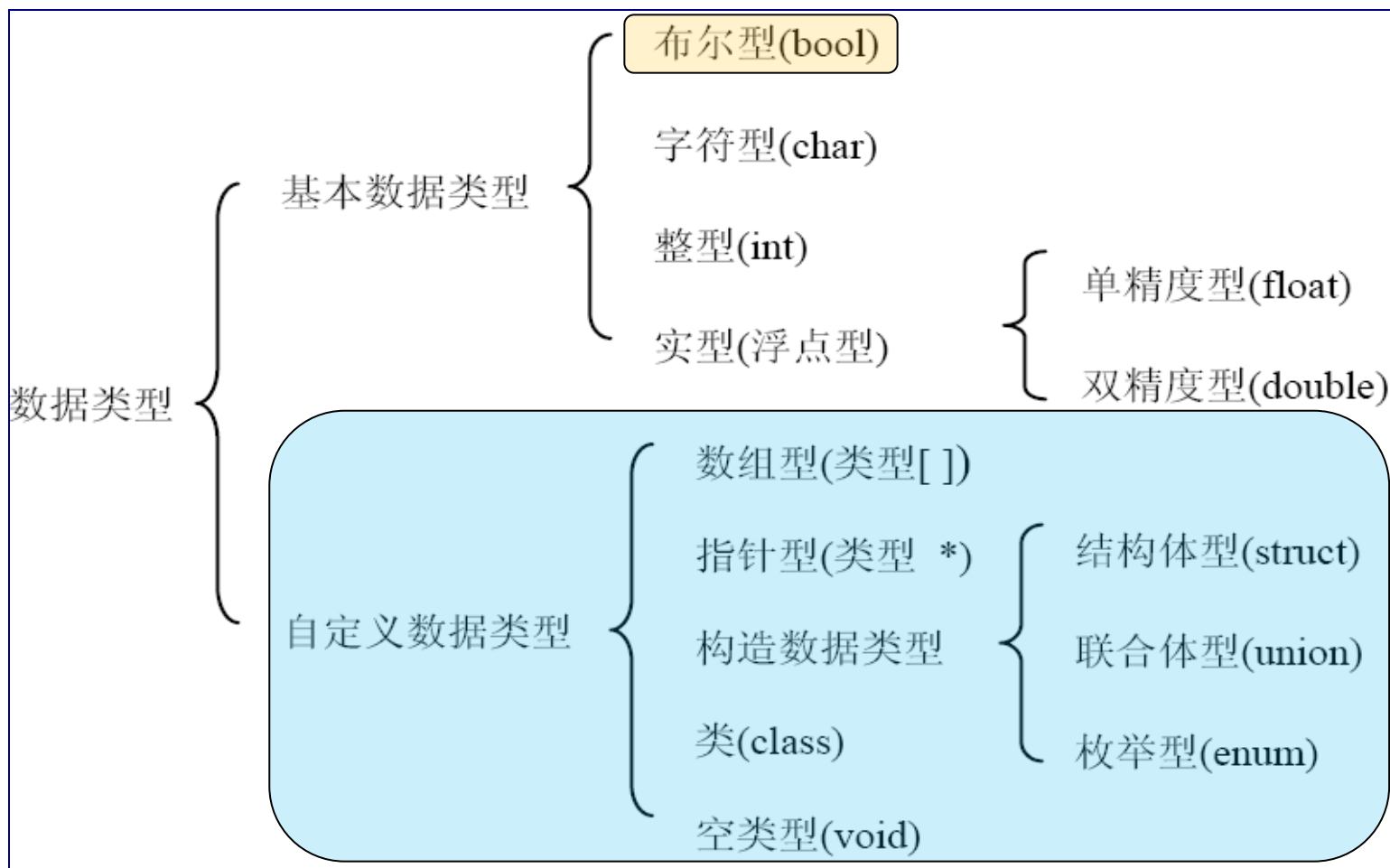
} **后使用变量**

← **输出结果**

运算对象与运算类型受控(2)

- 什么是运算对象
 - 带类型的数据
- C/C++ 有哪些类型的数据
 - 基本类型
 - 整形 (布尔类型)
 - 实型 (浮点型)
 - 字符型
 - 程序员 (用户) 定义类型
- 针对运算对象的运算类型受限
 - 算术运算
 - 关系 (逻辑运算)
 - 特殊运算

计算机能看懂的类型受控



C/C++的运算符受控

- 算术运算符 $+$ $-$ $*$ $/$ $\%$
- 关系运算符 $<$ $>$ $==$ $>=$ $<=$ $!=$
- 逻辑运算符 $!$ $\&\&$ $\|\$
- 位运算符 $>>$ \sim $|$ $^$ $\&$
- 条件运算符 $?$ $:$
- 求字节数运算符: `sizeof`
- 下标运算符 $[]$
- 赋值运算符 $=$
- 逗号运算符 $,$
- 指针运算符 $*$, $\&$
- 强制类型转换运算符: (类型)
- 分量运算符 $.$ \rightarrow

语句表达形式受控(3)

- 2大类语句

- 非可执行语句（主要2种）

- 新引入类型定义语句
 - 变量（常量）定义语句

- 可执行语句（主要三种）

- 赋值语句
 - 函数调用语句（含输入输出）
 - 控制语句
 - 分支
 - 循环

再看一个例子

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

```
int main()
```

```
{
```

```
char a;
```

非可执行语句，声明一个字符型变量

```
a = 'G';
```

可执行语句，给变量 a 赋值

```
cout << "猜猜我是哪个字母：" << endl;
cin >> a;
```

可执行语句
函数调用

```
if (a != 'G')
```

可执行语句
控制语句

```
cout << "你猜错了！" << endl;
```

```
return 0;
```

可执行语句
控制语句

可执行语句
函数调用

```
}
```

大小写判断

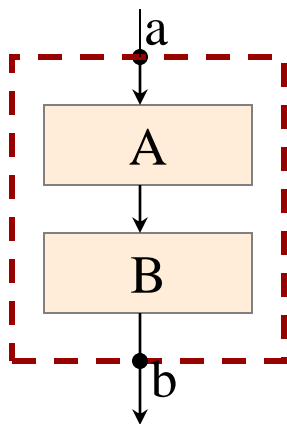
```
#include<iostream>
using namespace std;
int main()
{
    char a = ' ';
    cout << "输入一个符号：" << endl;
    cin >> a;
    if (a >= 'A')&&(a<= 'Z')
        cout << "输入是大写字母！" << endl;
    else if (a >= 'a')&&(a<= 'z')
        cout << "输入是小写字母！" << endl;
    else cout << "输入是其它符号！" << endl;
    return 0;
}
```

程序的逻辑结构受控(4)

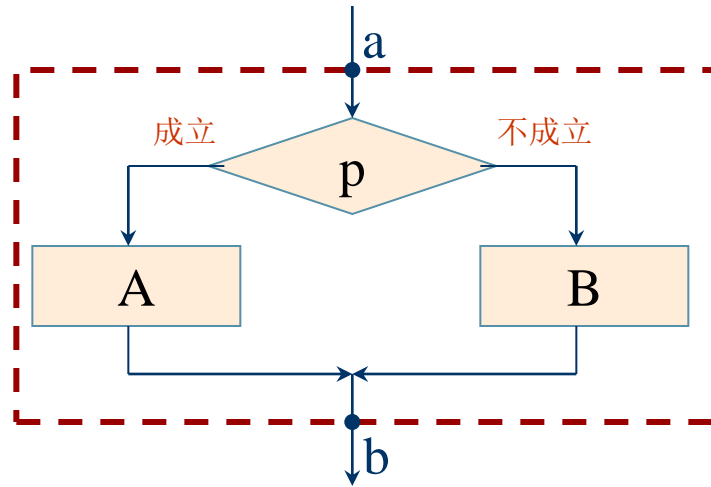
- **任何程序设计语言只需提供3种基本结构**
 - 纷繁复杂问题的求解均可由3种基本的逻辑结构表达

■ C. Bohm & G. Jacopini, "Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules," Communications of the ACM, vol9(5) May 1966, pp 366-371.

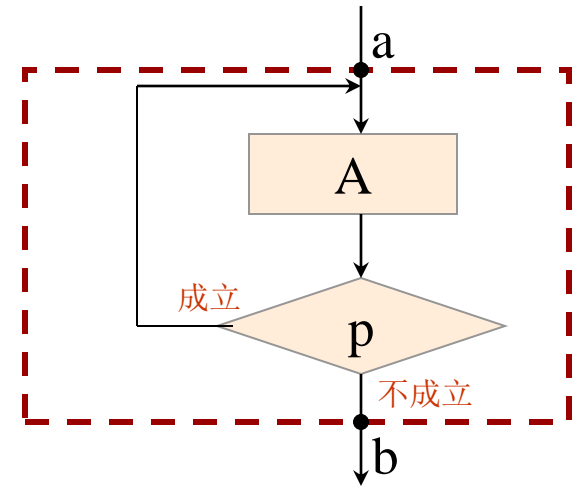
程序中的3种逻辑结构



顺序结构



分支结构



循环结构

■ 分支结构和循环结构还有不同的变种和表达形式

程序的逻辑结构：分支

- 解决问题时经常遇到选择情况

- 条件满足时：如何往下走

- 条件不满足时：如何往下走

- 例：解一元二次方程

$$ax^2 + bx + c = 0$$

当 $\Delta = b^2 - 4ac \geq 0$ 时, 有实数解.

判别式的不同取值（如 $\Delta \geq 0$ ）决定不同求解策略

再看例子：判断大小写

```
#include<iostream>
using namespace std;
int main()
{
    char a = ' ';
    cout << "输入一个符号：" << endl;
    cin >> a;
    if (a >= 'A')&&(a<= 'Z')
        cout << "输入是大写字母！" << endl;
    else if (a >= 'a')&&(a<= 'z')
        cout << "输入是小写字母！" << endl;
    else cout << "输入是其它符号！" << endl;
    return 0;
}
```

程序的逻辑结构：循环

- 在一个程序中，某种基本运算会连续重复多次
- 例：计算 $1+2+3+4+5$
- 步骤如下：
 - 步骤1：先计算 $0+1$ ，得到 $S \leftarrow 0+1$;
 - 步骤2：将上一步得到的结果加上 2 ，得到 $S \leftarrow 1+2=3$
 - 步骤3：将上一步得到的结果加上 3 ，得到 $S \leftarrow 3+3=6$
 - 步骤4：将上一步得到的结果加上 4 ，得到 $S \leftarrow 6+4=10$
 - 步骤5：将上一步得到的结果加上 5 ，得到 $S \leftarrow 10+5=15$,

如果计算 $1+2+3+\dots+1000$ 呢？

程序的逻辑结构：循环

- 计算 $1+2+\dots+100$

```
#include <iostream>
using namespace std;
int main()
{   int i,sum=0;
    for( i=1;i<=100;i++)
        sum = sum+i;
    cout<<sum<<endl;
    return 0;
}
```

内容

- 程序设计的基本思想
- 程序语言的受控特点
- 如何学好编程

如何学好程序设计语言

掌握

30几个关键字

十几种基本数据类型 + 30几个运算符

三种基本的逻辑语句（控制）



外加

输入输出的灵活使用

顺序结构程序

```
#include<iostream>
using namespace std;
int main()
```

```
{
```

```
    float a = 0, b = 0, temp = 0;
```

```
    cout << "Input a and b:"<<endl;
```

```
    cin >> a >> b;
```

```
    cout << "a = "<< a << ", b = " << b<<endl;
```

```
    temp = a; a = b; b = temp;
```

```
    cout << "a = " << a << ", b = " << b<<endl;
```

```
    return 0;
```

```
}
```

```
Input a and b:
3.14159 2.71828
a = 3.14159, b = 2.71828
a = 2.71828, b = 3.14159
```

理解什么叫初始化

执行顺序与书写顺序一致

分支结构程序

```
#include<iostream>
using namespace std;
int main()
```

```
{
```

```
    int x , y ;
```

此处未做初始化

```
    cin >> x >> y;
```

```
    if (x > y)
```

```
        cout << "Max number is: " << x << endl;
```

```
    else
```

```
        cout << "Max number is: " << y << endl;
```

```
    return 0;
```

```
}
```

```
123 321
Max number is: 321
```

执行顺序与书写顺序不一致

C/C++的输入输出

- C/C++自身没有输入输出语句，但提供了强大的库含有丰富的输入输出功能
- 库是什么
 - 函数/类的集合，典型的输入输出：
 - **C**: `stdio.h` 标准的输入输出函数集
 - **C++**: `iostream` 输入输出类库
- 库的使用方法
 - `#include` 引导库

<> 或 " " 两种形式

`#include <iostream>` //(只在标准库中找函数)

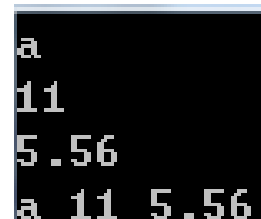
`#include "stdio.h"` //(先在当前位置找，找不到时，再在标准库中找)

C++中的输入输出

- 对应的库
 - `iostream.h`: 输入输出
- 常用函数（方法）
 - 输入: `cin`
 - 对应于标准输入（键盘）
 - 输出: `cout`
 - 对应于标准输出（显示器）

cin输入/cout输出举例

```
#include<iostream>
using namespace std;
int main()
{ char a;
  int b;
  float c;
  cin>>a>>b>>c;
  cout<<a<<" "<<b <<" "<<c<<endl;
  return 0;
}
```

A terminal window with a black background and white text. It shows the input sequence 'a', '11', and '5.56' on three separate lines. The output line shows 'a 11 5.56'.

跳过回车符

适用于不同数据类型的混合输入输出
endl 表示输出到此之后换到下一行

再看输入输出

```
#include<iostream>
using namespace std;
int main()
```

```
{
```

```
    float a = 0, b = 0, temp = 0;
```

```
    cout << "Input a and b:"<<endl;
```

```
    cin >> a >> b;
```

```
    cout << "a = "<< a << ", b = " << b<<endl;
```

```
    temp = a; a = b; b = temp;    // a和b交换值
```

```
    cout << "a = " << a << ", b = " << b<<endl;
```

```
    return 0;
```

```
}
```

```
Input a and b:
3.14159 2.71828
a = 3.14159, b = 2.71828
a = 2.71828, b = 3.14159
```

双引号表示的字符串原样输出

C语言的输入/出

- `#include <stdio.h>` //使用库
- 输入数据
 - 一般格式: `scanf` (格式控制、地址列表)
- 输出数据
 - 一般格式: `printf` (格式控制 , 输出列表)

`%d`:以带符号的十进制形式输出整数
`%o`:以八进制无符号形式输出整数(小写字母)
`%x`:以十六进制无符号形式输出整数

格式化输出

- 格式控制包括格式说明和普通字符：
 - 格式说明由“%”和格式字符组成，用于将输出数据转换成指定类型的数据和格式，如“%d”；
 - 普通字符，想要原样输出的字符。
- 输出列表：要输出的数据序列，可以是表达式。

例：假定 $a=3, b=4$ 则：

`printf("%d * %d = %d", a, b, a*b);`

格式说明

普通字符

$3 * 4 = 12$

输出结果：

格式符	意 义	格式符	意 义
%c ✓	字母	%d ✓	十进制整形数值
%i ✓	同 %d	%f ✓	浮点数
%e ✓	科学技术法表示的数（e表示）	%E	科学技术法表示的数（E表示）
%g ✓	%e和%f之间选择较短的	%G	%E和%f之间选择较短的
%o ✓	无符号8进制整数 (小写)	%s ✓	字符串
%u ✓	无符号整数	%x ✓	无符号16进制(小写x)整数
%X	无符号16进制(大写X)整数	%p ✓	输出(读)指针内容
%n ✓	显示（得到） 已输出（输入） 的字符数	%%	显示百分号

输出示例:

d	十进制整数	int a=567;printf ("%d",a);	567
x,X	十六进制无符号整数	int a=255;printf ("%x",a);	ff
o	八进制无符号整数	int a=65;printf ("%o",a);	101
c	单一字符	char a=65;printf ("%c",a);	A
s	字符串	printf ("%s", "ABC");	ABC
%%	百分号本身	printf ("%%%");	%

输入

- 功能：按格式从键盘上输入数据；
- 格式：scanf (格式控制、地址列表)
 - 格式控制与printf 相似，但地址列表只能表示变量地址，即：以&引导的变量，如 &a。
 - 如果地址列表有多个，则对应的多个输入数据之间的间隔符必须与格式部分的相应符号一致；如果输入时没有间隔符，空格符，回车符和Tab 键，都可作为当前数据结束符。
 - 前面表格带“√” 表示适用于 scanf。

格式控制的例子

- 2个输入例子

例子: `scanf("%d%d",&a,&b);`

键盘输入: 123 456, 得到 `a=123, b=456;`

例子: `scanf("%c%c",&c1,&c2);`

键盘输入: ab, 得到 `c1='a', c2='d';`

如何学好程序设计

掌握程序设计语言

具备程序设计技能

运用程序设计语言的能力

数据合理表示的能力

数据加工方法的能力

课堂思考1：变量 x/y 值交换

- 输入两个整数 x, y
- 交换 x 和 y 的值，并输出

用两种方法实现

变量 x/y 值交换

```
#include <iostream>
using namespace std;
int main()
{
    int x,y,z;
    cin>>x>>y;
    z=x;
    x=y;
    y=z;
    cout<<x<<y<<endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    int x,y;
    cin>>x>>y;
    x=x+y;
    y=x-y;
    x=x-y;
    cout<<x<<y<<endl;
    return 0;
}
```

节省一个变量

思考2：英文大小写转换方法

A: $(01000001)_2 / (65)_{10}$

<div>LH</div>	0000	0001	0010	0011	0100	0101	0110	0111
0000	NUL	DLE	SP	0	@	P	,	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	,	7	G	W	g	w
1000	BS	CAN)	8	H	X	h	x
1001	HT	EM	(9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

ASCII码

大写转小写

```
#include <iostream>
using namespace std;
int main()
{
    char c;
    cin>>c;
    c=c+32;
    cout<<c<<endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    char c;
    cin>>c;
    c=c-'A'+'a';
    cout<<c<<endl;
    return 0;
}
```

注意观察规律！
如果是小写转大写呢？

思考3： 计算最大值

- 语句形式一：
 - » 格式： `if (expression)`
`statement`
- 语句形式二：
 - » 格式： `if (expression)`
`statement-1`
`else`
`statement-2`
- 输入两个整数 `x,y`
- 分别用左边两种结构计算最大值

求最大值

```
#include <iostream>
using namespace std;
int main()
{
    int x,y,max;
    cin>>x>>y;
    if(x>y)
        max=x;
    else max=y;
    cout<<max<<endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    int x,y,max;
    cin>>x>>y;
    max=y;
    if(x>y)
        max=x;
    cout<<max<<endl;
    return 0;
}
```