NLP 南京大学 自然语言处理研究组 Natural Language Processing Group

南京大学 NANJING UNIVERSITY

程序中的数据（基础）
——部分基本数据类型及其操作

黄书剑

- 数据和数据类型
- 程序中的数据形式
- 数据的输入输出
- 数据的基本操作
- 良好的编程习惯

南京大学
NANJING UNIVERSITY

数据和数据类型

一个略微复杂的例子

- 例：计算一个正方形的周长（计量单位为米）。

```
#include <stdio.h>

int main( ){
    int di = 1;                          //di是一个整型变量，表示直径，初始值是1
    char mu = 'm';                       //mu是一个字符型变量，表示单位的英文符号
    scanf("%d", &di);                      //从控制台输入整数di的值，%d表示整型
    printf("The perimeter is: %d", 4 * di);          //输出计算结果
    printf("%c \n", mu);                 //输出计量单位
    return 0;
}
```

程序中的数据：值、标识符　　编译程序如何理解程序中的数据?

数据类型

- **数据是程序的一个重要组成部分，在程序中首先需要对要处理的数据进行描述。**
- **数据的描述是通过数据类型来实现的。一种数据类型由两个集合构成：**
  - 值集：规定了该数据类型能包含哪些值（包括这些值的结构）。
  - 操作（运算）集：规定了对值集中的值能实施哪些运算。
  - 例如：整数类型就是一种数据类型，它的值集：由整数构成，它的操作集：加、减、乘、除等运算。
- **区分数据类型的好处是便于实现对数据的可靠、有效处理。**

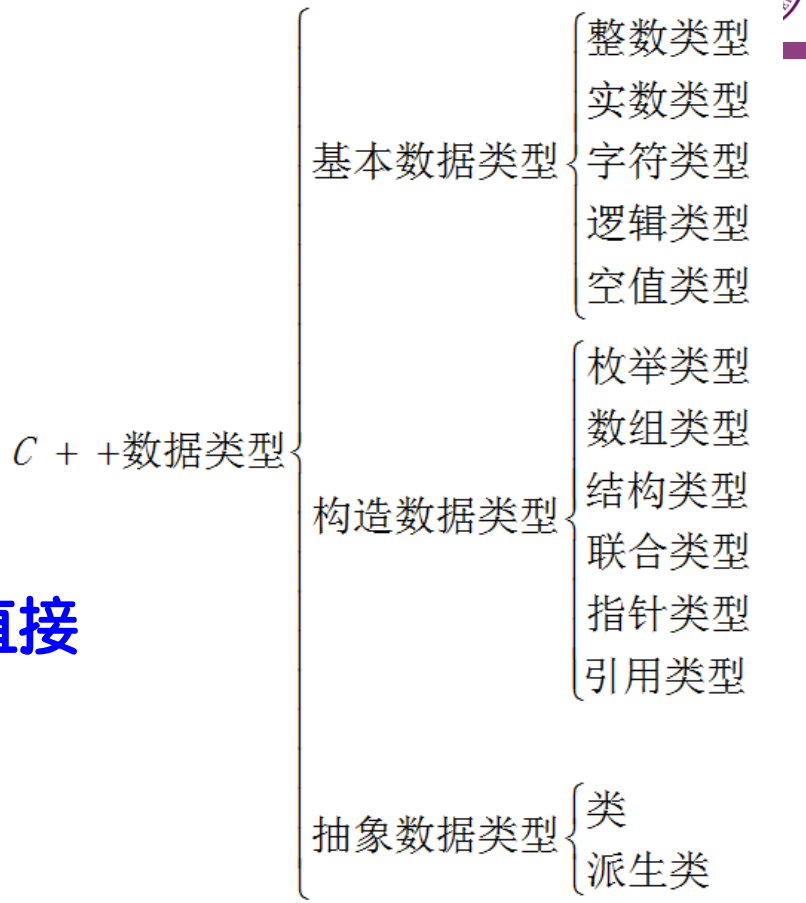
数据类型的分类

- **数据类型一般可以分为：**
  - 简单数据类型
    - 值集中的数据是不可再分解的简单数据。
    - 如：整数类型、实数类型等。
  - 复合数据类型
    - 值集中的数据是由其它类型的数据按照一定的方式组织而成。
    - 如：列表（由线性关系的元素组成）、矩阵（由具有行、列关系的元素组成）、向量（由分量组成）、学生信息（由学号、姓名、出生日期等组成）、…。

C/C++数据类型

- **基本数据类型**
  - 语言预先定义好的数据类型，常常又称为标准数据类型或内置数据类型（built-in types）。
  - 它们都是简单类型。
- **构造数据类型**
  - 利用语言提供的类型构造机制从其它类型构造出来的数据类型。
  - 它们大多为复合数据类型。
- **抽象数据类型(C++)**
  - 利用数据抽象机制把数据的表示对使用者隐藏起来的数据类型。
  - 它们一般为复合数据类型，在面向对象程序设计中用于描述对象。

- **部分C/C++基本数据型：**
  - 整数类型
  - 实数类型
  - 字符类型
  - 逻辑类型
  - 空值类型（void也是一种类型）
- **基本数据类型的数据能被机器指令直接进行操作。**



整数类型

- **整数类型通常用于描述整数。C/C++按精度提供以下整数类型：**
  - int
  - short int或short
  - long int或long

    “short int”的范围 ≤ “int”的范围 ≤ “long int”的范围
- **在计算机内部，上述整数类型的值采用固定长度的补码表示：**
  - short int占2个字节（-32768～32767 ）
  - long int占4个字节 （-2147483648～2147483647 ）
  - int占2个或4个字节，一般由计算机的字长决定。
- **C++新标准还提供了long long int（8个字节）**

超出表示范围：溢出 Overflow

实数类型

- **实数类型（又称浮点型），通常用于描述实数。C/C++根据精度把实数类型分为：**
  - float（单精度型）
  - double（双精度型）
  - long double（长双精度型）

    “float”的范围 < “double”的范围 ≤ “long double”的范围
- **在计算机内部，上述实数类型的值采用固定长度的浮点表示：**
  - float占4个字节（$-3.402823466X10^{38}$～$3.402823466X10^{38}$）
  - double占8个字节（$-1.7976931348623158X10^{308}$～$1.7976931348623158X10^{308}$）
  - long double占8个或10个字节（由具体的实现决定）

布尔类型（逻辑类型）

- **逻辑类型用于描述“真”和“假”这样的逻辑值，分别表示一个条件的满足和不满足。**
  - `_Bool` (C99)
  - `bool` (C++内建，C99 `stdbool.h`)
- **逻辑型变量的取值包括：**
  - `false` (整数 0)
  - `true` (整数 1)

字符类型

- 字符类型通常用于描述文字类型数据中的一个字符。
- 字符在计算机中存储的是它的编码。
  - `char`：表示单字节编码的字符。
  - `wchar_t`：表示多字节编码的字符。

常见的字符集及其编码

- **ASCII字符集**
  - 10个数字
  - 52个英文字母（包括大、小写）
  - 其它一些常用符号（如标点符号、数学运算符等）
  - 采用7位二进制编码表示（占用一个字节），可扩充成8位，最多表示256个字符。
  - 0~9十个数字、26个大写英文字母以及26个小写英文字母的编码各自是连续的。
- **在C/C++中用char类型描述ASCII字符。**

https://zh.wikipedia.org/wiki/ASCII

常见的字符集及其编码（续）

- **Unicode（国际通用字符集）**
  - 包含大部分语言中的字符
  - 2～4个字节编码
  - C++用wchar_t描述
- **Big5（繁体中文字符集）**
  - 包含台湾、香港繁体汉字字符
  - 2个字节编码
  - C++用2个char描述
- **GB2312（简体中文字符集）**
  - 包含中文简体和部分繁体汉字字符
  - 2个字节编码
  - C++用2个char描述
- **Shift-JIS（日文字符集）**
  - 包含日语汉字、假名字符
  - 2个字节编码
  - C++用2个char描述

南京大学
NANJING UNIVERSITY

程序中的数据形式

数据的形式

- **常量**
  - 用于表示在程序执行过程中不会改变或不允许被改变的数据。
    - 例如：闰年的天数、圆周率等。
- **变量**
  - 用于表示在程序执行过程中可变的数据。
    - 例如：用户输入的正方形边长，其他部分的计算结果等。

- **此处不同的数据形式是从数据是否可以改变的角度出发，每个具体数据仍然对应不同的数据类型**

字面常量（literal constant）

- **程序中直接书写的常量**
  - 整数
    - 如：7，3276500L，4352U
    - 整数字面常量默认为int类型，如果需要使用long int或unsigned int，可以使用L或者U进行制定
  - 实数
    - 如：3.14， 456.78， -0.0057，4.5678E2，-5.7e-3，456.78f
    - 实数字面常量默认为double类型，如果需要使用float或者long double，可以只用字母F或者L制定

字面常量（literal constant）

- **程序中直接书写的常量**
  - 字符常量
    - 如'm'，'\n'（换行符），'\r'（回车符），'\t'（制表符）
    - 可以是单个字符、转义字符等，也可以通过编码和转义指定字符
    - 字符常量的类型为char
  - 字符串常量
    - 如 "Hello World!", "This is a string", "This is a two-line \n message!"
    - 字符串常量中可以使用的单个字符同字符常量的规定
    - 字符串常量的类型为一维字符数组常量（构造数据类型）

使用字面常量的例子

- 例：计算一个圆的周长和面积（计量单位为米）。

```
#include <stdio.h>

int main( ){
    int r = 1;                          //r是一个整型变量，表示半径，初始值是1
    char mu = 'm';                       //m是一个字符型变量，表示单位的英文符号
    scanf("%d", &r);                     //从控制台输入整数d的值，%d表示整型
    printf("The perimeter is: %f%c \n",2 * 3.14 * r, mu);   //输出周长
    printf("The area is: %f%c2 \n",3.14 * r * r, mu);       //输出面积
    return 0;
}
```

使用常量的例子

- 例：计算一个圆的周长和面积（计量单位为米）。

```
#include <stdio.h>
#define PI 3.14
int main( ){
    int r = 1;                          //r是一个整型变量，表示半径，初始值是1
    char mu = 'm';                      //m是一个字符型变量，表示单位的英文符号
    scanf("%d", &r);                    //从控制台输入整数d的值，%d表示整型
    printf("The perimeter is: %f%c \n", 2 * PI * r, mu);   //输出周长
    printf("The area is: %f%c2 \n", PI * r * r, mu);       //输出面积
    return 0;
}
```

符号常量定义后可以在程序中反复使用，其值皆为定义时指定的值

符号常量（manifest constant）

- **对于程序中多次使用的字面常量，可以定义成符号常量。即通过常量定义给常量取一个名字，在程序中通过常量名来使用这个字面常量。**
- **定义格式：**
  - `#define` <符号常量> <字面常量>
    - 其中<符号常量>是一个标识符
    - 如： `#define PI 3.1415926`
- **使用：**
  - 后续程序中可以通过符号常量标识符来使用这个字面常量
  - 如： `PI * 2 * r`

关于符号常量的说明

- **预处理命令**
  - #开头的命令是预处理命令，将在编译前作为前处理执行
  - #include （将被引用文件拷贝到当前文件中），#define
  - 预处理命令行的末尾一般不加分号
- **#define**
  - 实际是将源程序语句中的符号常量全部替换成相应的字面常量，参加编译的是替换后的字面常量。
  - 因此，符号常量并不显示指定类型，其类型由字面常量决定
  - *除了符号常量外，#define可以定义表达式等更复杂的内容，称为宏定义（Macros）
    - `#define ADD(a, b) a + b`

用符号代替直接量/表达式是一种抽象！
也是一种对信息的存储/记忆！

使用符号常量的好处

- **保证程序对常量使用的一致性**
  - 3.14、3.1416、3.1415926
- **增加程序的易读性**
  - #define PASS_SCORE 60
  - #define MINUTES_PER_HOUR 60
- **增强程序的易维护性**
  - 通过改变符号常量，可以统一改变程序中参与计算的数据
- ***bool类型的值true和false可以看成是C++语言预定义的两个符号常量，它们的值分别为1和0**

提升程序的一致性是一个重要原则（v.s. copy&paste）

使用字面常量的例子

- 例：计算一个圆的周长和面积（计量单位为米）。

```
#include <stdio.h>

int main( ){
    int r = 1;                          //r是一个整型变量，表示半径，初始值是1
    char mu = 'm';                       //m是一个字符型变量，表示单位的英文符号
    scanf("%d", &r);                     //从控制台输入整数d的值，%d表示整型
    printf("The perimeter is: %f%c \n",2 * 3.14 * r, mu);   //输出周长
    printf("The area is: %f%c2 \n",3.14 * r * r, mu);       //输出面积
    return 0;
}
```

变量

- **在程序执行过程中可变的数据称为变量。**
  - 程序中的部分数据可能要发生改变
  - 变量可以用于存储和标识这些数据

- **例如：在计算圆周长的式子“2*PI*r”中，**
  - 半径r就是一个可变的数据，它可能是通过用户输入得到，也可能由程序的其它部分计算得到，在程序运行过程中可能仍然会发生变化，比如由用户再次进行输入等。

try：能否写一个不使用变量的程序？这个程序将发生什么样的变化？为什么？

变量的存在使得程序有了可变的记忆功能，这使其能力变得更加强大

变量的基本属性

- **名字**
  - 标识符，区别不同的变量，用于在后续程序中访问对应的变量
- **类型**
  - 确定变量的值集以及操作集
- **值**
  - 可取值集中的任何一个值
- **内存空间和地址**
  - 存储变量值的内存空间以及该空间的地址。
  - 一般由编译系统自动管理，但C/C++提供基于内存地址的操作方式

变量的定义

- **C/C++语言规定：程序中使用到的每个变量都要有定义（有的语言不需要）。变量定义格式为：**

<类型名> <变量名>;

或者 <类型名> <变量名>=<初值>;

或者 <类型名> <变量名>(<初值>); // 仅C++，与对象初始化一致

例如：

```
int a=0; //int a(0);
int b=a+1; //int b(a+1);
double x;
```

或：

```
int a=0,b=a+1;
//同类型变量可以写在一起，
//用‘,’分开
double x;
```

变量的赋值（assignment）与初始化（initialization）

- **通过给变量赋值，可以使变量获得有意义的值，还可以使变量的值在程序运行过程中发生改变。**
  - C/C++中用 = 表示赋值，将右边的值存入左边变量。
- **变量的值可以在定义的时候赋一个初值，即初始化（在程序编译期间进行赋值），也可以后期赋值（在程序执行期间进行赋值任务）**
  - `int n, d = 2;`
  - `d = d + 1;`
- **如果定义的变量未初始化（uninitialized），则其值可能不确定，可能会给后面的相关计算带来预想不到的结果。**
- **Tips:**
  - 尽可能在定义变量的同时初始化该变量，如果变量的引用处和其定义处相隔比较远，变量的初始化很容易被忘记。

另一种表示不可变数据的方法

- **加const修饰的变量（常量）**
  - **const** <类型名> <常量名> **=** <值>**;**

  - 例如： const double PI=3.1415926;
- **与变量具有相同的基本属性，但其值在程序运行中不可以更改**
  - const修饰的变量应在定义时赋初值

回顾

- **数据类型**
  - 值集、操作集
- **字面常量、符号常量**
  - 不说明类型，由编译器自动推断
- **变量**
  - 定义时声明类型，使用时注意初始化
  - 变量名、类型、值（状态）、内存空间和地址
  - 常量
- **数据的基本属性：类型、值（状态）、内存空间和地址**

逻辑层面　　物理层面

一个略微复杂的例子

- 例：计算一个正方形的周长（计量单位为米）。

```
#include <stdio.h>

int main( ){
    int di = 1;                          //di是一个整型变量，表示直径，初始值是1
    char mu = 'm';                       //mu是一个字符型变量，表示单位的英文符号
    scanf("%d", &di);                      //从控制台输入整数di的值，%d表示整型
    printf("The perimeter is: %d", 4 * di);          //输出计算结果
    printf("%c \n", mu);                 //输出计量单位
    return 0;
}
```

程序中的数据：值、标识符　　编译程序如何理解程序中的数据?

南京大学
NANJING UNIVERSITY

数据的输入输出简介

程序与外设

- **程序运行过程中往往需要与外部设备进行交互**
  - 输出指定的信息（print hello world）
  - 获取指定的信息（等待用户选择等）
- **直接与外部设备交互是困难的**
  - 涉及到多种不同设备的驱动、数据传输、缓存等一系列问题
- **因此，与外部数据的交互往往通过预先定义的库函数来进行**
  - C语言中支持格式化输入输出函数 scanf和printf
  - C++中实现了面向对象的输入输出 cin和cout

更多详细内容请参见教材第十章第一、第二节。

数据的输出

- 输出（output），一般是指将程序执行的结果显示到显示器上
- C语言调用函数库中的printf等输出函数，需要在程序头部用 #include <stdio.h> 包含输出函数的说明信息
- C语言的输出函数与具体数值类型有关，如：
  - printf("Hello World! \n"); 表示原样输出双引号中的字符串内容
  - printf("The s: %f", s); 表示输出浮点数s的值
  - printf("%c \n", mu); 表示输出字符mu的值并换行

更详细的格式细节可以查阅相关资料： https://en.cppreference.com/w/c/io/fprintf
https://www.runoob.com/cprogramming/c-function-printf.html

数据的输出

- 输出（output），一般是指将程序执行的结果显示到显示器上
- C++语言使用cout对象和插入操作符<<，需要在程序头部用#include <iostream>包含输入类的说明信息，并加上使用std名空间的说明。
- C++语言的输入自动适配数值类型(通过插入操作符的重载实现)
  - cout<<"Hello World! \n"; 表示原样输出双引号中的字符串内容
  - cout<<"The s: "<< s; 表示输出浮点数s的值
  - cout<< mu << " " <<endl; 表示输出字符mu的值并换行

更详细的介绍可以查阅相关资料： https://en.cppreference.com/w/cpp/io/cout
https://www.runoob.com/cplusplus/cpp-basic-input-output.html

数据（变量值）的输入

- **输入input，一般是指将需要的数据从键盘输入内存**
- **C语言调用函数库中的scanf等函数，需要在程序头部用#include <stdio.h>包含输入库函数的说明信息**
- **C语言的输入函数与具体数值类型有关，且参数为地址（&）**
  - scanf("%d", &n); 表示从控制台输入一个整数值
  - scanf("%f", &s); 表示从控制台输入一个实数值
  - *注意，scanf中的格式化字符串表明需要输入的内容
    - scanf("Please input %d", &r); //需要输入Please input 10 才能获取输入的数字10
    - scanf("%d,%d", &r, &d); //表示输入整数之间用,分隔

更详细的格式细节可以查阅相关资料： https://en.cppreference.com/w/c/io/fscanf
https://www.runoob.com/cprogramming/c-function-scanf.html

数据（变量值）的输入

- 输入input，一般是指将需要的数据从键盘输入内存
- C++语言使用cin对象和抽取操作符>>，需要在程序头部用#include <iostream>包含输入类的说明信息，并加上使用std名空间的说明。
- C++语言的输入自动适配数值类型(通过抽取操作符的重载实现)
  - int n; cin >> n; 表示从控制台输入一个整数值
  - float f; cin >> f; 表示从控制台输入一个实数值
    - 无法像scanf一样指定输入格式串
    - cin >> r >> d; 多个输入之间用空白符分隔

更详细的介绍可以查阅相关资料： https://en.cppreference.com/w/cpp/io/cin
https://www.runoob.com/cplusplus/cpp-basic-input-output.html

*语言兼容性问题

- C语言不支持面向对象输入输出
- C++中可以使用 <cstdio> 与 C 的输入输出保持兼容

- C语言中，scanf，sscanf，fscanf 等函数直接向指定内存地址进行输入，在内存使用上安全性较差
  - C11标准中引入了更为安全的版本：
    - scanf_s, sscanf_s, fscanf_s
- 但是，在C++标准的 <cstdio> 中并未支持前述改进
- 但是，VS2015中进行C++开发时会对不安全的版本报错！

南京大学
NANJING UNIVERSITY

数据的基本操作

操作符（运算符）

- **操作符用于对数据进行运算，这里的数据称为操作数，它们可以是：**
  - 常量、变量、函数调用的结果、其它操作符的运算结果
- **例如，在下面的计算式子中，**

```
a+4
x=b
a/f(10)
(-a)*(b-c)
```

  - +、-（减法）、-（取负）、*、/、f（函数调用）以及 =（赋值）都是操作符
  - a、b、4、c、10、x以及(-a)、(b+c)、f(10)都是操作数

算术操作符

- **算术操作符用于实现通常意义下的数值运算（操作数类型一般为算术型），包括：**
  - 取负“-”与取正“+”，例如：-x
  - 加“+”、减“-”、乘“*”、除“/”
  - 整除“/”用于整型操作数时表示整除，小数点后面的数将舍去，并且一般不进行四舍五入；浮点数参与运算时表示浮点数除法
    - 例如：3/2的结果为1；-10/3的结果为-3
  - 取余数“%”用于计算两个整型数相除的余数。
    - 例如：10%3的结果为1；8%2的结果为0
    - “%”的操作数一般不为负数，a%b 的结果按 a - (a/b)*b 解释

算术操作符（续）

- 自减“--”和自增“++”
  - 单目操作符，把操作数（只能是变量）减（或加）1
  - 可以前置，也可以后置。例如，++x和x++
  - 前置与后置的区别是：

```
int x=1,y;
y = (++x)   //x的值是2，y的值为2（先加后用）
y = (x++)   //x的值是2，y的值为1  （先用后加）
```

- **算术操作的结果，其类型一般与操作数类型相同，因此，可能会产生“溢出”等问题。**

关系操作符

- 程序中经常要根据某个条件是否满足来决定其后续的动作，这里的条件往往体现为对数据的比较操作。
- 关系操作符用于对数据进行大小比较，包括：

  `>` （大于），`<` （小于），`>=` （不小于），`<=` （不大于），

  `==` （相等），`!=` （不等）
- 操作数通常为算术类型。关系操作的结果为`bool`类型的值：`true`或`false`。例如：

  `3 > 2` 的结果为`true`

  `4.3 < 1.2` 的结果为`false`

  `'A' < 'B'` 的结果为`true`

  `false < true` 的结果为`true`

逻辑操作符

- 操作应满足的条件也可以表示成对多个条件的逻辑运算（并且、或者等），用于复杂条件的描述。逻辑运算操作符包括：
  - ! (逻辑非)
  - && (逻辑与)
  - || (逻辑或)
- 逻辑操作的操作数为bool类型，通常为关系操作的结果。例如：
  - `!(a > b)` //或者，`a <= b`
  - `(age > 10) && (age < 30)` //错误写法：`10 < age < 30`
  - `(ch < '0') || (ch > '9')`
- 逻辑操作的结果为bool类型。

- **逻辑运算真值表**

```
!true  -> false
!false -> true

false && false -> false
false && true  -> false
true  && false -> false
true  && true  -> true

false || false -> false
false || true  -> true
true  || false -> true
true  || true  -> true
```

其他操作符

- **条件操作符（?:） d1?d2:d3**
  - d1表示条件，如果d1的值为true或非零，则运算结果为d2，否则为d3。
  - 例如：　　c = (a>b)?a:b　//a和b中的大者赋值给c
- **逗号操作符　d1,d2,d3,...**
  - 从左至右依次进行各个运算，操作结果为最后一个运算的结果。
  - 逗号操作表示的计算更加清晰。
  - 例如，把 z = a+b+c+d　　表示成：x = a+b, y = c+d, z = x+y （z的值的含义更加清晰）
- **sizeof操作符：**

  sizeof(<类型名>**)**　或　**sizeof(**<表达式>**)**
  - 计算某类型的数据占用的内存大小（字节数）

操作符的优先级和结合性

- **一个表达式中包含的多个操作符的运算需要规定！**
- **对相邻的两个操作符，按下面规则确定：**
  - 圆括号：圆括号内的先运算。例如：a*(b-c)，先算“-”
  - 优先级：优先级高的先运算。例如：a+b*c，先算“*”
  - 结合性：相同优先级按左结合或右结合。例如： a+b-c，先算“+”；a=b=c，先算第二个“=”
- **对不相邻的操作符，C++一般没有规定计算次序（&&、||、?:和,操作符除外）**
  - 例如，对于：(a+b)*(c-d)，C++没有规定+和-的计算次序！

回顾

- **对数据可以进行的操作由数据类型决定！**
  - 操作符：基本操作
    - 算术运算、关系运算、逻辑运算
  - 复合操作：表达式

- **多个操作注意考虑优先级和结合性！**

静态类型语言和动态类型语言

- 一个数据，不管是常量还是变量，它都属于某种类型！
- 静态类型语言（statically typed languages）
  - 在程序中必须为每个数据明确指定一种类型。
  - 程序通常采用编译方式执行。
- 动态类型语言（dynamically typed languages）
  - 在程序运行中数据被用到时才确定它们的类型。
  - 程序通常采用解释方式执行。
- C/C++是静态类型语言

静态类型语言的好处

- **静态类型语言的好处：**
  - 提高程序的可靠性，便于编译程序自动进行类型一致性检查。
  - 便于产生高效的可执行代码。
- **例如，对于“x+y”，根据x和y的数据类型，编译程序就能在程序运行之前**
  - 知道它的合法性
  - 自动进行类型转换
  - 生成合适的机器指令

*渐进类型语言(Gradual Typing) 有兴趣的同学可以自行了解一下

南京大学
NANJING UNIVERSITY

良好的编程习惯

良好的编程习惯

- 设计**正确**的算法、数据结构与代码
- 采用**适合**计算机的算法、**合理**组织数据
- 考虑周全、引入故障检测
- 顾及系统、平台的差异，避免歧义
- 合理抽象、分解、组合
- 提高程序的易读性
  - 注意程序的排版
  - 为程序书写注释
  - 注意自定义标识符的命名风格
  - …

好的程序：

正确 (correct)
高效 (efficient)
可靠 (reliable)
可移植 (portable)
可重用 (re-usable)
可扩展 (Scalable)
易读 (readability)
……

***课件有时没有遵循所建议的规则，这是为了将相关内容放在一张幻灯片上，便于讲解。**

排版

- **C程序的书写比较自由，不必在规定的行或列 书写规定的内容。**
- **不过良好的书写格式不仅可以使程序美观，还有利于提高程序的可读性，便于程序的调试和维护。**
- **初学者应注意养成良好的书写习惯，比如：**
  - 一行只写一个语句
  - 采用好的缩进模式（即在同一块语句前插入等量的空格-用Tab键，并保持前后一致）
  - 在操作符两端、逗号后恰当地添加空格
  - 在程序段落之间恰当地添加空行
  - …

良好的程序设计风格

```
#include <stdio.h>
int main()
{
   printf("Now Join Us! \n");
   return 0;
}
```

√

```
#include <stdio.h>
int main(){
   printf("Now Join Us! \n");
   return 0;
}
```

√ 一种更紧凑的表示方法

```
#include <stdio.h>
int main(){  printf("Now Join Us! \n");
}
```

应该用缩进清晰表示函数结果

关于自定义标识符命名规则

- **存在多种不同的命名风格**
  - 程序员倾向于使用其个人的命名约定，而不喜欢别人规定他们如何编写代码
  - 然而，通用稳定的命名约定更利于团队合作和日后自己阅读
- **驼峰式大小写（Camel Case）**
  - `firstName、lastName`
- **匈牙利表示法(Hungarian Notation)**
  - 以标准的3或4个字母前缀来表示变量的数据类型，比如表示学生年龄的整型变量就应该命名为intStuAge.)

`int_stu_age`

https://zh.wikipedia.org/wiki/%E9%A7%9D%E5%B3%B0%E5%BC%8F%E5%A4%A7%E5%B0%8F%E5%AF%AB

https://zh.wikipedia.org/wiki/%E5%8C%88%E7%89%99%E5%88%A9%E5%91%BD%E5%90%8D%E6%B3%95

自定义标识符命名建议☆

【总则】采用一致的、有意义的标识符名字。对不同种类的标识符最好采用不同风格的名字。

【建议1】自定义标识符应当直观，用词尽量准确，可望文知意。切忌使用汉语拼音简拼来命名。

【建议2】标识符的长度应当符合“min-length && max-information”原则。

一般来说，长名字能更好地表达含义，但名字并非越长越好；单字符的名字也是有用的，常见的如i, j, k, m, n, x, y, z等，它们通常可用作函数内的局部变量。

自定义标识符命名建议☆

【建议3】程序中不要出现仅靠大小写区分的相似的标识符。

例如：

```
int  x, y, X;// 变量x 与 X 容易混淆
void foo(int x);    // 函数foo 与FOO容易混淆
void FOO(int y);
```

【建议4】用一对反义词命名具有相反含义的变量或函数等。

例如：

```
int minValue, maxValue;
int SetValue(…), GetValue(…);
```

自定义标识符命名建议☆

【建议5】 函数名和类型名用大写字母开头的单词组合而成。

例如：

```
void  Init(void);
void  SetValue(int value);
```

系统定义的类型名、main函数名及库函数名除外

【建议6】 变量名和参数名的首单词用小写字母开头。

例如：

```
int flag;
int stuAge;
int current_value
```

自定义标识符命名建议☆

【建议7】习惯使用符号常量，符号常量名全用大写字母，用下划线分割单词。

例如：

```
#define MAX_LENGTH  100
#define PI 3.14
```

小结

- **教学要求：**
  - 了解基本数据类型的基本原理
  - 了解数据在程序中的形式
- **实践：**
  - 掌握基本数据类型及其运算操作
  - 能够利用控制台进行数据的输入输出交互
  - 了解并培养良好的编程风格
- **阅读：教材第二章、第十章相关内容（其中关于数的计算机内部表示可以暂时跳过，后续再行讨论）**