



南开大学
Nankai University

Nankai
University

ISSUE 4
WINTER 2021

C++中的数据

龚成 南开大学软件学院

cheng-gong@nankai.edu.cn

<https://en.nankai.edu.cn>

NANKAI



数据存储与编码

起始地址

数据长度

存储内容

编码格式



起始地址

- 计算机中的内存是线性编码的，每一个数据和指令都一起存储在内存中的某一位置，并且以某一地址作为起始地址。
- 在32位计算机/程序中，起始地址是一个32位的无符号整数。

```
#include <iostream>
```

Local variable's address: 0x16dc8f008

Array's address: 0x16dc8f010

```
int main() {  
    int localVar = 42;  
    int myArray[5] = {1, 2, 3, 4, 5};  
  
    std::cout << "Local variable's address: " << &localVar << std::endl;  
    std::cout << "Array's address: " << myArray << std::endl;  
  
    return 0;  
}
```



数据长度

- 数据需要占用一定的存储空间。
- 现代计算机使用二进制进行数据表示和存储，其基本单位是位（binary）。
- 位是一个极小的单位，表示一个0/1值。现代计算机体系结构中通常以8位为一个单位，表示一个字节（Byte）。



数据长度

- C++中，我们通常说某种变量占用多大内存，指的是字节数，比如int通常具有4B这么大。
- 数据的长度对应是数据的内存占用，指的是从数据起始地址开始，存储数据的所需要的连续的内存占用空间。
- C++中可以使用sizeof符号或者对应类型或者变量的内存占用大小（为什么对应类型也可以看大小呢？）。



存储内容

- 计算机的内存是一维线性连续的，确定起始地址和数据长度，就可以确定数据的存储位置，获取存储内容。
- 现代计算机使用二进制表示数据，任意数据和指令的存储内容都是二进制的
- 数据。
- 那么怎么区分不同的数据呢？



编码格式

- 编码格式指我们怎么编码数据以及解码对应的二进制存储内容。
- C++中使用变量类型来隐式地指明数据的编码、解码和存储的格式和算法。
- 因此，C++中的数据类型非常重要。



基本数据类型 与派生类型

整型

浮点型

派生关键字

基本数据类型

- 写程序主要是对数据进行计算或处理，也就是使用 C++ 语言支持的数据类型和运算，完成计算任务。
- 如下的两个程序分别列出来两种不同的计算程序，分别涉及了计算符号和数据类型的定义。

course1 >  sum_test.cpp > ...

```
1 //sum_test.cpp
2 #include<iostream>
3 using namespace std;
4 int main()
5 {
6     int a,b,sum;
7     a = 43;
8     b = 37;
9     sum = a + b;
10    cout << "The sum is " << sum;
11    cout << endl;
12    return 0;
13 }
```

course1 >  area_of_cicle.cpp > ...

```
1 //program 3-2.cpp
2 #include<iostream>
3 using namespace std;
4 int main()
5 {
6     const float pai = 3.14;
7     float radius;
8     cout << " Enter radius:";
9     cin >> radius;
10    float area = pai * radius * radius;
11    cout << "\n The area of circle is ";
12    cout << area << endl;
13    return 0;
14 }
```



基本数据类型

- 类型的信息决定了变量值的表示方法以及所需的内存量，类型的信息还决定了相关的算术运算的意义。
- 类型概念的几个要点是：
 - 每一项数据应惟一地属于某种类型。
 - 每一数据类型意味着一个有明确定义的值的集合。
 - 同一类型的数据占用相同大小的存储空间。
 - 同一类型的数据具有相同的（允许对其施加的）运算操作集。
- C++程序中的数据类型可以分为如下几类：

表 3.1 C++语言中类型的划分

系统提供	基本类型	int、float、double、char、bool、void
	派生类型	(修饰符+基本类型)
用户定义	完全由用户定义	class, (struct, union)
	部分由用户定义	enum (int 类型的子集)
由其他类型导出		array、struct、pointer、reference



基本数据类型

- 基本类型是具有下面 3 个特征的数据类型：
 - 由系统定义和提供。
 - 它们是构造所有其他类型的原始出发点。
 - 它们是几乎所有程序设计语言都包含的数据类型。
- C++语言的基本数据类型有：int 型、float 型、double 型、char 型、bool型和 void 型。

- C++中的整型指的是可以精确表示的整数，其对应于数学中的自然数集合，但是受限于位宽，整型的表示范围通常有限。
- 整型不只可以表示整数，还可以表示离散的状态等等。因此，C++中将整型进一步划分为以下类型：
 - int
 - bool
 - char
 - 枚举
 - 指针
 - ...

整型



- `int` 型又称整型。是最常用最基本的数据类型。
- **范围：**原则上是所有整数，实际的值集为计算机所能表示的所有整数。这个范围是有限的，因此程序员在编程时必须经常考虑因数据过大而溢出的问题。
- **存储：**占用的存储空间按不同的计算机和编译系统而有所差别，目前在PC 机上运行的各种 C++语言规定 `int`型数据占用 4 B即 32 bit 空间。
- **运算：**`int` 型数据允许算术运算、关系运算等许多种运算。

整型



- char 型又称字符型，即把单个字符作为一种数据处理。
- char 型的值集是全部基本字符、ASCII 码集或扩充的 ASCII 码集对应的全部符号。
- char 型的数据占用 1 B即 8 bit 空间。
- 在作为数字计算的时候，char 型数据与低位宽的 int 型等价，因此可参加的运算相当广泛。
- 字符集可与单字节整数有完整的对应关系（ASCII 码），因此还可把 char 型看做是可以用来表示单字节整数的字符型。

ASCII码



- 计算机本身不能直接区分不同的字母、数字或特殊符号，它是根据每个符号对应的编码来识别这些基本符号的。这些符号的编码表称为ASCII码表。
- ASCII 是美国标准信息交换码（American Standard Code for Information Interchange）的英文缩写，ASCII 码表把 95 个基本（可打印）符号和 33 个控制字符共 128 个字符与 7 位二进制数 0000000~1111111 共 128 个数码建立了对应关系，实际上任何一个基本符号在计算机内的表示形式就是这样一个二进制数码。

表 2.1 ASCII 码表

代码	字符	名称	代码	字符	名称	代码	字符	名称	代码	字符	名称
000	NUL	无效	032		空格	064	@		096	'	单引
001	SOH	标题始	033	!	叹号	065	A		097	a	
002	STX	正文始	034	"	双引	066	B		098	b	
003	ETX	正文尾	035	#	#号	067	C		099	c	
004	EOT	传递止	036	\$	币号	068	D		100	d	
005	ENQ	查询	037	%	百分号	069	E		101	e	
006	ACK	信号确认	038	&	与号	070	F		102	f	

ASCII码



表 2.1 ASCII 码表

代码	字符	名称	代码	字符	名称	代码	字符	名称	代码	字符	名称
016	DLE	换码	048	0		080	P		112	p	
017	DC1	设备控制	049	1		081	Q		113	q	
018	DC2	设备控制	050	2		082	R		114	r	
019	DC3	设备控制	051	3		083	S		115	s	
020	DC4	设备停机	052	4		084	T		116	t	
021	NAK	信息否认	053	5		085	U		117	u	
022	SYN	同步	054	6		086	V		118	v	
023	ETB	块传送止	055	7		087	W		119	w	
024	CAN	作废	056	8		088	X		120	x	
025	EM	纸用完	057	9		089	Y		121	y	
026	SUB	置换	058	:	冒号	090	Z		122	z	
027	ESC	换码	059	;	分号	091	[方括号	123	{	括号
028	FS	文件分离	060	<	小于	092	\	反斜杠	124		竖线
029	GS	分组	061	=	等于	093]	方括号	125	}	括号
030	RS	记录分离	062	>	大于	094	^		126	~	波纹
031	US	元素分离	063	?	问号	095	_	下横线	127	DEL	删除

整型



- `bool` 型在新版的 C++语言中被列为基本类型，它只有两个值：`false`、`true`，表示逻辑的真和假，可参加逻辑运算和作为逻辑表达式及关系表达式的结果。`bool`型的存储空间为1B，即8bit。
- `bool`只能参与逻辑运算。
- `bool`本身只有0/1两种表示，实际上只需要1bit存储就可以了，为什么在C++中也要将`bool`存储为8bit的值呢？
- `void` 型称为无值型。`void` 型是一种较为抽象的概念，在 C++语言中它用来说明函数及其参数，没有返回值的函数说明为 `void` 类型的函数，没有参数的函数其形参表由 `void` 表示。
- 有了 `void` 类型，C++语言规定，所有函数说明（`main` 函数除外）都必须指明（返回）类型，都必须包含参数说明。

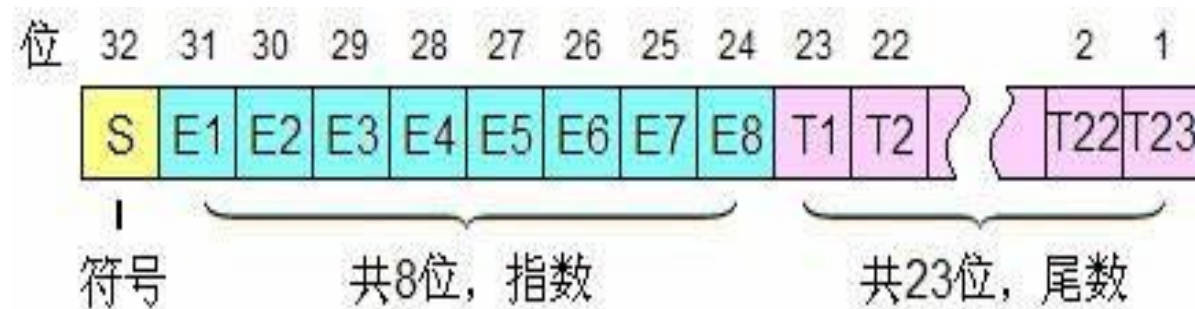


浮点型

- C++中的浮点型对应于数学中的实数类型，因此在许多书和编程语言中浮点又被称为实数型。但是受限于表示位宽，浮点的表示范围和精度通常有限。
- C++中实现了IEEE754标准的浮点类型：
 - float
 - double
- 浮点表示通常是不精确的，比如1.3这个实数通常难以在计算机中获得精确的1.3的表示，因此对浮点数做操作时，通常用作数学计算，不做条件判定。

浮点型

- float 型又称浮点型，它对应着数学中的实数概念，带小数点的数。
- float 型的值集，原则上是任意大小和精度的小数，实际的值集虽然不可能是任意大小，但由于采用尾数+阶码的表示形式，所以其可表示浮点数的范围可大到 $\pm 3.4 \times 10^{38}$ ，表示的精度可以小到 1.0×10^{-38} 。因此，在一般的应用问题中，float 型数据总是可以满足精度和大小的要求，不会出现溢出现象。
- float 型数据一般占用 4 B，即 32 bit 空间。当精度较高或数值较大时，人们往往使用 double 型（占用 8 B，即 64 bit 空间）。
- float 型数据与 int 型数据的区别在于它们所参加的运算操作类型是不同的。比如增量运算（++，--），条件判断等语句中。





派生类型及关键字

- 基本类型经过简单的字长及范围放大或缩小，就形成了基本类型的简单派生类型。
- 派生类型的说明符由 `int`、`float`、`double`、`char` 前面加上类型修饰符组成。类型修饰符包括：
 - `short`: 短的，缩短字长。
 - `long`: 长的，加长字长。
 - `signed`: 有符号的，值的范围包括正负值。
 - `unsigned`: 无符号的，值的范围只包括正值。



派生类型及关键字

表 3.2 基本类型及派生类型的字长及值域

类 型 名	字长 (B)	取 值 范 围
bool		{false,true}
char	1	-128 ~ 127
(signed) char	1	-128 ~ 127
unsigned char	1	0 ~ 255
short (int)	2	-32 768 ~ 32 767
(signed) short (int)	2	-32 768 ~ 32 767
unsigned short (int)	2	0 ~ 65 535
int	4	-2 147 483 648 ~ 2 147 483 647
(signed) int	4	-2 147 483 648 ~ 2 147 483 647
unsigned (int)	4	0 ~ 4 294 967 295
long (int)	4	-2 147 483 648 ~ 2 147 483 647
(signed) long (int)	4	-2 147 483 648 ~ 2 147 483 647
unsigned long (int)	4	0 ~ 4 294 967 295
float	4	-3.4E38 ~ 3.4E38
double	8	-1.7E308 ~ 1.7E308
long double	10	-1.1E4932 ~ 1.1E4932
void	0	{ }



数据类型的内存量

Code

type_demo.cpp

```
1 // helloworld.cpp
2 #include "type_demo.h"
3
4 int main(void)
5 {
6     sizeoftype();
7     return 0;
8 }
```

type_demo.h

```
1 // helloworld.cpp
2 #include <iostream>
3 using namespace std;
4
5 void sizeoftype(){
6     cout<<"sizeof(int)="<<sizeof(int)<<endl;
7     cout<<"sizeof(float)="<<sizeof(float)<<endl;
8     cout<<"sizeof(double)="<<sizeof(double)<<endl;
9     cout<<"sizeof(wchar_t)="<<sizeof(wchar_t)<<endl;
10    cout<<"sizeof(char)="<<sizeof(char)<<endl;
11    cout<<"sizeof(bool)="<<sizeof(bool)<<endl;
12 }
```

Run

```
sizeof(int)=4
sizeof(float)=4
sizeof(double)=8
sizeof(wchar_t)=2
sizeof(char)=1
sizeof(bool)=1
```



常量与变量

全局变量

局部变量

外部变量

常量变量

生存期与作用域

存储类型属性

变量



- 变量是数据在程序中出现的主要形式，在变量第 1 次被使用前它应被说明。变量说明的格式为：

[<存储类>] <类型名或类型定义><变量名表>;

```
static long sum;
```

- 类型名或类型定义在任何变量说明语句中都必须包含不可缺省。
- 变量名表可连续定义多个并赋初始值

变量名表: <变量名>[=<表达式>], <变量名表>

```
int size, high, temp= 37;
```




全局变量

- 其说明语句不在任何一个类定义、函数定义和复合语句（程序块）之内的变量。
- 全局变量所占用的内存空间在内存的数据区，在程序运行的整个过程中位置保持不变。

```
4  int global_a = 0;
5  int main(void)
6  {
7      int local_a = 1;
8      sizeof type();
9      cout<<"global_a="<<global_a<<" , local_a="<<local_a<<endl;
10     return 0;
11 }
```

局部变量



- 其说明语句在某一类定义、函数定义或复合语句之内的变量。简单地说，说明语句包含在某一对分割符“{”和“}”之内的变量为局部变量，否则为全局变量。
- 局部变量占用的空间一般位于为程序运行时设置的临时工作区，以堆栈的形式允许反复占用和释放。

```
4  int global_a = 0;
5  int main(void)
6  {
7      int local_a = 1;
8      sizeof type();
9      cout<<"global_a="<<global_a<<" , local_a="<<local_a<<endl;
10     return 0;
11 }
```

外部变量



- `extern`: 把变量说明为外部变量。
- 一个变量被说明为外部变量，其含义是告诉系统不必为其分配内存，该变量已在这一局部的外面定义。
- 外部变量一般用于由多个文件组成的程序中，有些变量在多个文件中被说明，但却是同一变量，指出某一变量为外部变量就避免了重复分配内存。
- 可以在程序中多次说明`extern`变量，但是为`extern`变量赋初值会报警，但是并不会出错，只是等价于`extern`失效。

```
4 // int global_a = 0;  
5 int global_a=0;  
6 extern int global_a;  
7 extern int global_a;
```

常量变量

- C++程序中的数据可分为**常量**（constant）与**变量**（variable）两大类。在程序执行过程中其值不能被改变的数据称为常量，其值可以改变的数据称为变量。常量又分**有名常量**和**字面常量**。
- **字面常量**的类型是根据书写形式来区分的。例如，475、200 是 int 型，3.1416、200.0 是float 型，‘a’、‘4’、‘@’ 是 char 型。
- **有名常量**和**变量**在程序中必须遵循“**先声明，后使用**”的原则，程序中出现的所有有名常量和变量都必须在使用前由常量说明语句和变量说明语句进行说明。

```
const float pai = 3.1416;
```




常量变量

- 常量说明语句的格式必须以**关键字** `const` 开头为：

const <类型名><常量名> = <表达式>;

const float pai = 3.1416;

- 表达式的值应与该常量类型一致的表达式（常量和变量也是表达式）

`char* str = "Hello C++";`  `const char* str = "Hello C++";`
warning: ISO C++ forbids converting a string constant to 'char*'

- 常量和变量都要求系统为其分配内存单元，所以可以把有名常量视为一种**不允许赋值改变的或只读不写**的变量，称为 `const` 变量。
- 与宏定义的区别：宏定义也能定义常量，不过，用宏替换的方法定义符号常量与 `const` 方式的实现机制是不同的：
 - 宏替换是在编译时把程序中出现的所有标识符 `N` 或 `pai` 都用 `1000` 和 `3.1416` 来**替换**，这里并没有一个只读不写的 `const` 变量存在；
 - 宏替换的方式中没有类型、值的概念，仅是两个字符串的代换。



字面常量

整型

浮点型

字符型

字符串型

字面常量



- C++程序中的常量是指固定不变的量，有两种表示形式：一种称为有名常量，一种称为字面常量（literal constant）。例如圆周率`float pai=3.1416`，其中 `pai`就是一个有名常量，`pai`是量 `3.1416` 的名字，而 `3.1416` 称为字面常量。
- 字面常量分为 4 类：`int` 型常量、`float` 型常量、`char` 型常量和字符串常量。字面常量是我们可以输入程序的量，简单来说，如果我们想给程序输入内容，只能是这4种类型的量。

course1 >  helloworld.cpp > ...

```
1  // helloworld.cpp
2  #include <iostream>
3  using namespace std;
4  int main(void)
5  {
6      cout<<"Hello World!"<<endl;
7      return 0;
8  }

8  for(chicken=0; chicken<100; chicken+=3)
9      for(hen=0; hen<=33; hen++)
10         if((cock=100-chicken-hen)> -1)
```


course1 >  cin_test.cpp > ...

```
1  //program2-3.cpp
2  #include <iostream>
3  using namespace std;
4  int main(void)
5  {
6      int myage;
7      cout<<"My age is ";
8      cin>>myage; //输入年龄(一个整数)
```

整型



- int 型常量即整型常量，实际上就是整数。

course1 >  literal_constant.cpp > ...

```
1  // literal_constant.cpp
2  #include <iostream>
3  using namespace std;
4  int main(void)
5  {
6      int radiu = 4;
7      float pai = 3.1415926535;
8      float weight = 6.9e4;
9      cout<<" radiu = "<<radiu<<"\n pai = "<<pai<<"\n weight = "<<weight<<"g\n";
10     return 0;
11 }
```

```
$ g++ literal_constant.cpp -o literal_constant.exe
$ ./literal_constant.exe
radiu = 4
pai = 3.14159
weight = 69000g
```


浮点型



- float 型常量即浮点常量，浮点常量有两种表示法：
 - 小数点表示法：4.75、2.0、-473.385。
 - 科学表示法：1.2e35、-7.37e-3

course1 >  literal_constant.cpp > ...

```
1  // literal_constant.cpp
2  #include <iostream>
3  using namespace std;
4  int main(void)
5  {
6      int radiu = 4;
7      float pai = 3.1415926535;
8      float weight = 6.9e4;
9      cout<<" radiu = "<<radiu<<"\n pai = "<<pai<<"\n weight = "<<weight<<"g\n";
10     return 0;
11 }
```

```
$ g++ literal_constant.cpp -o literal_constant.exe
$ ./literal_constant.exe
radiu = 4
pai = 3.14159
weight = 69000g
```

字符型



- char 常量即字符常量，如：'A'、'g'、'3'、'!'。C++语言还定义了一些特殊的字符常量，全是用反斜杠“\”开头的，如下(要记住)

表 2.3 字符常量的表示及含义

字 符 常 量	ASCII 码	含 义	符 号
\a	007	响铃	BEL
\b	008	退格	BS
\f	012	走纸	FF
\n	010	换行	LF
\r	013	回车	CR
\t	111	水平制表	HT
\v	011	垂直制表	VT
\\	092	反斜杠	\
\'	044	单引号	'
\"	034	双引号	"
\?	063	问号	?
\0	048	整数 0	NUL
\DDD	0DDD	八进制整数	
\xHHH	0xHHH	十六进制整数	

字符串型



- 字符串常量是用双引号括起来的字符序列，称为字符串常量。比如“string constant”。

course1 > helloworld.cpp > ...

```
1 // helloworld.cpp
2 #include <iostream>
3 using namespace std;
4 int main(void)
5 {
6     cout<<"Hello World!"<<endl;
7     return 0;
8 }
```

course1 > literal_constant.cpp > main(void)

```
1 // literal_constant.cpp
2 #include <iostream>
3 using namespace std;
4 int main(void)
5 {
6     int radiu = 4;
7     float pai = 3.1415926535;
8     float weight = 6.9e4;
9     const char* str = "Hello World! \nHello China! \0 Hello Nankai\n";
```

course1 > cin_test.cpp > ...

```
1 //program2-3.cpp
2 #include <iostream>
3 using namespace std;
4 int main(void)
5 {
6     int myage;
7     cout<<"My age is ";
8     cin>>myage; //输入年龄(一个整数)
9     cout<<endl;
10    return 0;
11 }
```



字符串型

- 一个字符串常量是一个特殊的字符序列或字符数组，其长度为该字符串中所有字符的个数加 1。原因是除了保存串中字符（包括空格）之外，**在最后存一串尾符'\0'**。例如，字符串常量"Pascal"的长度为 7。
- 字符串中也可包括特殊字符，如下。不过在使用特殊符号'\0' 时，会产生问题，系统将把它作为串尾符而忽略了它后面的其他字符。

```
course1 > G+ literal_constant.cpp > main(void)
```

```
1  // literal_constant.cpp
2  #include <iostream>
3  using namespace std;
4  int main(void)
5  {
6      int radiu = 4;
7      float pai = 3.1415926535;
8      float weight = 6.9e4;
9      const char* str = "Hello World! \nHello China! \0 Hello Nankai\n";
10     cout<<" radiu = "<<radiu<<"\n pai = "<<pai<<"\n weight = "<<weight<<"g\n";
11     cout<< str<<endl;
12     return 0;
13 }
```

```
$ g++ literal_constant.cpp -o literal_constant.exe
$ ./literal_constant.exe
radiu = 4
pai = 3.14159
weight = 69000g
Hello World!
Hello China!
```



随机数生成

代码示例



随机种子设置与随机数生成

```
#include <iostream>
#include <random>

int main() {
    std::mt19937 generator(0);
    int n;
    char s;
    std::cin>>n>>s;
    if (s == 'i') {
        std::uniform_int_distribution<int> distribution(0, 100);
        for (int i = 0; i < n; ++i) {
            std::cout << distribution(generator) << std::endl;
        }
    } else if (s == 'f') {
        std::uniform_real_distribution<float> distribution(0.0, 1.0);
        for (int i = 0; i < n; ++i) {
            std::cout << distribution(generator) << std::endl;
        }
    } else {
        std::cout<<"error"<<std::endl;
        return 0;
    }
    return 0;
}
```