2. Variables

3. Values and Variables

Values and Variables

- Numeric value
  (Character, string)
- Variables
- Declarations
- Assignment
- Identifiers
- Reserved words

Integer Values (1)

```
#include <iostream>

int main() {
   std::cout << 4 << "\n";
   std::cout << "4" << "\n";
   std::cout << '4' << "\n";
   std::cout << "4\n";
   std::cout << 4/2 << "\n";
   std::cout << "4/2" << std::endl;
   std::cout << "4/2" << '\n‘;
}
```


Integer Values (2)

```
#include <iostream>

int main() {
   std::cout << 5/2 << "\n";
   std::cout << 5/2*2 << "\n";
   std::cout << 2*5/2 << "\n";
   std::cout << -3000000000 << "\n"; // -2,147,483,648~2,147,483,647
   std::cout << 1'234 << "\n";       // modern C++
                                     // line comment – annotation
                                     // ignored by compilers
}
```

Variables and Assignment (1)

```
#include <iostream>
int main() {
   int x;      // variable declaration of a integer type
   x = 10;     // assignment, 10 = x;
   std::cout << x << '\n';
   std::cout << "x" << '\n';
}
```

```
#include <iostream>
int main() {
   int x;
   x = 10;
   std::cout << x << '\n';
   x = 20;
   std::cout << x << '\n';
   x = 30;
   std::cout << x << '\n';
}
```


Variables and Assignment (2)

```
#include <iostream>
int main() {
   int x = 10; // variable declaration & assignment (initialization)
   std::cout << x << '\n';
   std::cout << "x" << '\n';
}
```

```
#include <iostream>
int main() {
   int x{10};
   int y(20);
   std::cout << x << '\n';
   std::cout << y << '\n';
}
```

Variables and Assignment (3)

```
int x, y, z;

int x = 0, y, z = 5;

int x = 0;
int y;
int z = 5;

int a, b;       // a and b are separate memory locations
a = 2;
b = 5;
a = b;          // a and b are not same memory location
b = 4;          // a?, b?
```


Identifiers (1)

- An identifier is a word used to name things.
- Variables, functions, classes, …
- Identifiers
  - Identifiers must contain at least one character.
  - The first character must be an alphabetic letter (upper or lower case) or the underscore (ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_)
  - The remaining characters (if any) may be alphabetic characters (upper or lower case), the underscore, or a digit (ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_0123456789)
  - No other characters (including spaces) are permitted in identifiers.
  - A reserved word cannot be used as an identifier

Identifiers (2)

- Reserved words (keywords)
  - alignas decltype namespace struct alignof default
  - new switch and delete noexcept template
  - and_eq double not this asm do not_eq
  - thread_local auto dynamic_cast nullptr throw
  - bitand else operator true bitor enum
  - or try bool explicit or_eq typedef break export
  - private typeid case extern protected typename
  - catch false public union char float register unsigned
  - char16_t for reinterpret_cast using char32_t friend
  - return virtual class goto short void compl if
  - signed volatile const inline sizeof wchar_t constexpr
  - int static while const_cast long static_assert
  - xor continue mutable static_cast xor_eq

Additional Integer Types (1)

- `short int ≤ int ≤ long int ≤ long long int`
- `unsigned short ≤ unsigned ≤ unsigned long ≤ unsigned long long`

- `short int (short) 2bytes -32,768~32,767`
- `int, 4bytes, -2,147,483,648~2,147,483,647`
- `long int (long), 4bytes -2,147,483,648~2,147,483,647`
- `long long int (long long), 8bytes -9,223,372,036,854,775,808~9,223,372,036,854,775,807`

Additional Integer Types (2)

- unsigned short int (unsigned short), 2bytes
  0~65,535
- unsigned int (unsigned), 4bytes
  0~4,294,967,295
- unsigned long int (unsigned long), 4bytes
  0~4,294,967,295
- unsigned long long int (unsigned long long), 8bytes
  0~18,446,744,073,709,551,615

- int x = 4456;
- long x = 4456L;
- long x = 4456l;

Floating-point Types

- float, 4bytes, 1.17549×10^{-38}, $3.40282\times10^{+38}$, 6digits
- double, 8bytes, 2.22507×10^{-308}, $1.79769\times10^{+308}$, 15digits
- long double, 8bytes, 2.22507×10^{-308}, $1.79769\times10^{+308}$, 15digits

- float x = 3.14f;
- double y = 6.022e23; // 6.022E23, 6.022e-23

```
#include <iostream>
int main() {
   double pi = 3.14159;
   std::cout << "Pi = " << pi << '\n';
   std::cout << "or " << 3.14 << " for short" << '\n';
}
```

Constant

- `const double PI = 3.14159;` `// once declared and initialized, // a constant may not be reassigned`

```
#include <iostream>
int main() {
  const double avogadros_number = 6.022e23, c = 2.998e8;
  std::cout << "Avogadro's number = " << avogadros_number << '\n';
  std::cout << "Speed of light = " << c << '\n';
}
```


Characters (1)

- `char ch = 'A';` ~~`// ch = "A";`~~

- The char data type is used to represent single characters: letters of the alphabet (both upper and lower case), digits, punctuation, and control characters (like newline and tab characters).
- Most systems support the American Standard Code for Information Interchange (ASCII) character set. Standard ASCII can represent 128 different characters.

```
#include <iostream>
int main() {
   char ch1, ch2;
   ch1 = 65;
   ch2 = 'A'+1;
   std::cout << ch1 << ", " << ch2 << ", " << 'A' << '\n';
}
```

Characters (2)

0	*null*	16	32	*space*	48	0	64	@	80	P	96	`	112	p
1		17	33	!	49	1	65	A	81	Q	97	a	113	q
2		18	34	"	50	2	66	B	82	R	98	b	114	r
3		19	35	#	51	3	67	C	83	S	99	c	115	s
4		20	36	$	52	4	68	D	84	T	100	d	116	t
5		21	37	%	53	5	69	E	85	U	101	e	117	u
6		22	38	&	54	6	70	F	86	V	102	f	118	v
7	*bell*	23	39	'	55	7	71	G	87	W	103	g	119	w
8	*backspace*	24	40	(	56	8	72	H	88	X	104	h	120	x
9	*tab*	25	41	)	57	9	73	I	89	Y	105	i	121	y
10	*newline*	26	42	*	58	:	74	J	90	Z	106	j	122	z
11		27	43	+	59	;	75	K	91	[	107	k	123	{
12	*form feed*	28	44	,	60	<	76	L	92	\	108	l	124	|
13	*return*	29	45	-	61	=	77	M	93	]	109	m	125	}
14		30	46	.	62	>	78	N	94	^	110	n	126	~
15		31	47	/	63	?	79	O	95	_	111	o	127	

Characters (3)

- '\n'—the newline character
- '\r'—the carriage return character
- '\b'—the backspace character
- '\a'—the "alert" character (causes a "beep" sound or other tone on some systems)
- '\t'—the tab character
- '\f'—the formfeed character
- '\0'—the null character (used in C strings, see Section 11.2.6)

- '\\'.
- std::cout << "C:\\Dev\\cppcode" << '\n';

Enumerated Types (1)

- `enum Color { Red, Orange, Yellow, Green, Blue, Violet };`

```
#include <iostream>
int main() {
   enum Color { Red, Orange, Yellow, Green, Blue, Violet };
   std::cout << Red << Orange << Yellow << std::endl;

   enum Animal { Cat = 1, Dog, Puppy=2 };
   std::cout << Cat << Dog << Puppy << std::endl;
}
```


Enumerated Types (2)

- `enum class Shade { Dark, Dim, Light, Bright };`
- `enum class Weight { Light, Medium, Heavy };`
- `Shade color = Shade::Light;`
- `Weight mass = Weight::Light;`

- `std::cout << (int)color << std::endl;`

Type Inference with auto

- `auto count = 0;`
- `auto ch = 'Z';`
- `auto limit = 100.0;`

```
// auto x;
```