中国民航大学 Civil Aviation University of China

Computer Programming

Sino-European Institute of Aviation Engineering

Module 2

Types Operators and Expressions

Outline

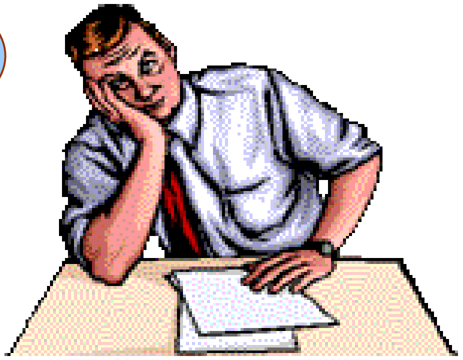
Problem

There is a circle swimming pool, we want to decorate it. So circumference needed.

Mathematical formula:

$$C = 2\pi r$$

How to calculate C with program?



Problem

How to implement mathematical formula?

expression

operand +operator

How to input and store data?

constant

variable

operator

How to process data?

Data, Data Types and Sizes

Program → **data storage**, **data process**

- **Expression**
  - **Operators**
    - Arithmetic
    - Assignment
    - Relational (comparison)
    - Logical
    - Others
  - **Operands**
    - Constants
    - Variables
    - Function call

Data, Data Types and Sizes

- **Data** – > **the processed object of a program.**

```
sum = 1+2;        printf("%d",sum);
sum = n1 + n2;
```

- **Data type** – > **define data structure, data size, data operation**

```
char c = 'a';    chat *p = "abc";
int i = 2;       int d = 3/i;
```

Data, Data Types and Sizes

- **Data type 数据类型**
  - **Basic types（基本类型）**
    - **int（整型）**
    - **char（字符型）**
    - **float**, **double**（浮点型）
  - **Pointer type（指针类型）**
  - **Construction types（构造类型）**
    - **array（数组）**
    - **struct （结构）**
    - **union （联合）**
    - **enum （枚举）**
  - **void（空类型）**

Data, Data Types and Sizes

- Different data type has different storage Size
- Same data type on different Platform has different storage Size
- Just use sizeof(data type) to get it.

```
sizeof (int);    sizeof(char);  sizeof(float); sizeof(double);
```

Constants and Variables

Constants

- its value *can not be changed* during program execution.

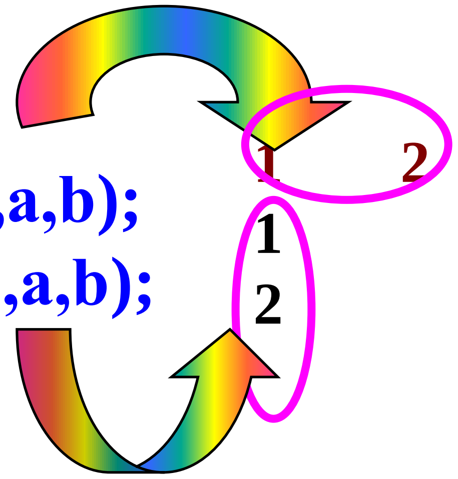
- Integer constant
  - Decimal 100，125，-100，0
  - Octal 011, 015, 026
  - Hexadecimal 0x38 ,0X1A
- Floating-point constant （实型常量）
  - Decimal point 3.14， 0.125，-3.789
  - Exponent 1e3 1.8e-3
-
  - Character constant （字符） ‘A’ ‘g’ ‘0’ ‘2’ ‘+’ ‘#’
  - String constant（字符串） “abc” , “ I love C”
- Symbolic constant（符号常量）

Character Constant

- Some characters can not be represented by ‘ ‘, so **escape sequence(转义字符) provided.**

\n	newline
\t	tab
\\	backslash
\’	single quote
\0	null
\r	Enter
\”	double quote
\b	backspace
\ddd	octal
\xhh	decimal

```
printf(“%d \t %d \t”,a,b);
printf(“%d \n%d \n”,a,b);
```



if a=1，b=2

’a’=’\141’ ’A’=’\101’ newline ’ \12’
’a’=’\ x61’ ’A’=’\x41’ newline ’ \xa’

Symbolic Constant

Question: the radius of a circle is provided, calculate its circumference and area.

```
void main ( )
{ float r, c, a;
   scanf (“%f”, &r);
   c = 2 * 3.1415926 * r ;
   a = 3.1415926 * r * r;
   printf(“c= %6.2f, a=%6.2f \n”, c, a);
}
```

something to replace constant 3.1415926?

Symbolic Constant

#define PI 3.1415926

Symbolic Constant

```
#define PAI 3.1416      /* define symbolic constant */
void main()
  {
     float r,c,a;
     scanf("%f",&r);
     c=2*PAI*r;              /* When compiling,PI be replaced by 3.1416 */
     a=PAI*r*r;                  /* When compiling,PI be replaced by 3.1416
    */
     printf("c=%6.2f,s=%6.2f\n",c,a);
  }
```

macro substitution

Symbolic Constant

```
#define   ONE      1
#define   COM      “company”
#define   MAX      100
#define   TWO      ONE+ONE
```

```
a= b+2;
printf (“%s”, “company”);
int array[100];
```

$\Rightarrow$

```
a= b+ TWO;
printf(“%s”,COM);
int  array[MAX];
```

String Constant

- A **string constant** is a sequence of zero or more characters surrounded by double quotes (**“**).

“	T	h	i	s		i	s		C		s	t	r	i	n	g	.	”

length =17

“	a	”

length=1

“		”

length=1

“	”

length=0

“	\	t	\	“	N	a	m	e	\	\	A	d	d	r	e	s	s	\	n	”

15

“	\	“	C		i	s		v	e	r	y		e	a	s	y	.	\	”	”

17

“	H	e		s	a	i	d	\	“	o	k	.	\	”	\	n	”

13

Variables and Declaration

NAME & name different

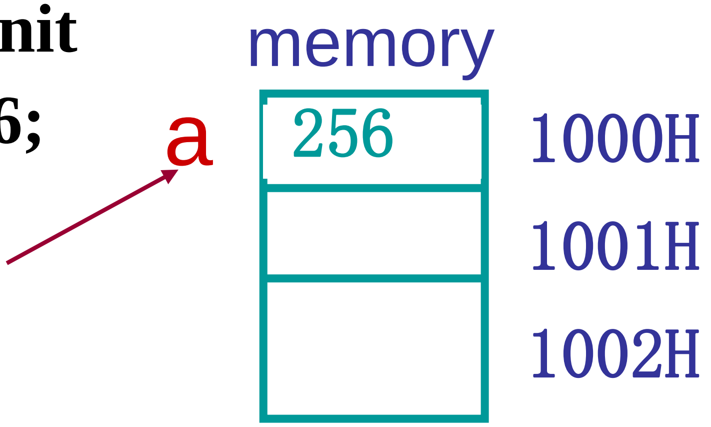
- Variable name ← Identifier

  declaration form: data type variable name;

  Examples: int a ; char b;

- Variable -> Memory Unit

  a = 8; a = 12; a = 256;



variable

- variable must be declared before used

Variables and Declaration

- A variable can be initialized in its declaration.

Form: **data type variable name =expression ;**

constant expression

Examples: **int a =20 ; char b = ‘c’;**

#define MAX 100

int x=0;

int y= MAX+1;

Integral Variable

- Integral data type, size and range

type	size	range
int	2bytes	-32768～32767
short	2bytes	-2^{15}～2^{15}-1
long	4 bytes	-2^{31}～2^{31}-1
unsigned	2 bytes	0～65535
unsigned short	2 bytes	0～65535
unsigned long	4 bytes	0～（2^{32}-1）

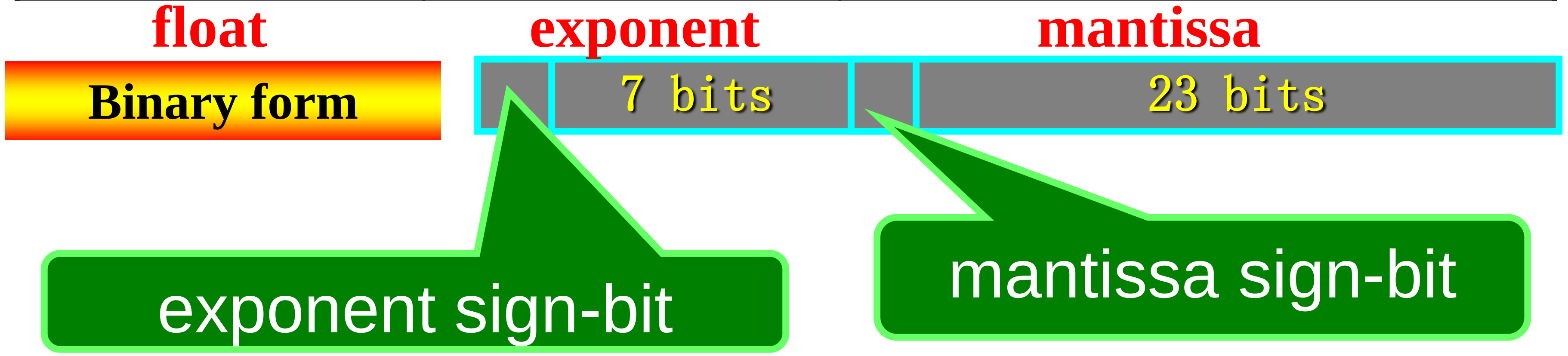
Integral Variable

- A program to add two numbers:

```
/* add2.c  */
void main()               /* add two numbers together */
{
    int n1,n2,s;     /* variable declaration */
    n1 = 32767;     /* Initialize a */
    n2 = 3;
   s = n1 + n2;       /* add n1 and n2 */
   printf(“s=%d\n",s);       /* output s */
 }
```

Float Point Data

Type	Memory size	range
float	4 bytes	$10^{-38} \sim 10^{38}$
double	8 bytes	$10^{-308} \sim 10^{308}$
long double	16 bytes	$10^{-4931} \sim 10^{4932}$



Example: 0.123456×10^{-2}

Decimal form

-	2	+	0.123456

Float Point Data

- Display a & b on the screen

```
void main( )
 {
     float a;
     double b;
     a=12345.6789;
     b=0.1234567891234567899e15;
     printf("a=%f,b=%f\n",a,b);
}
```

???

result

a=12345.6789，b=0.1234567891234567899e15

a=12345.678711，b=123456789123456.797000

Character Data

- The char variable is stored as its ASCII code

type	size	range
char	1	0~255

```
void main ( )
{  char c;
c = 'c';
printf("%c\n",c);
printf("%d\n",c);
}
```

'a' ⟷ 97

0 1 1 0 0 0 0 1

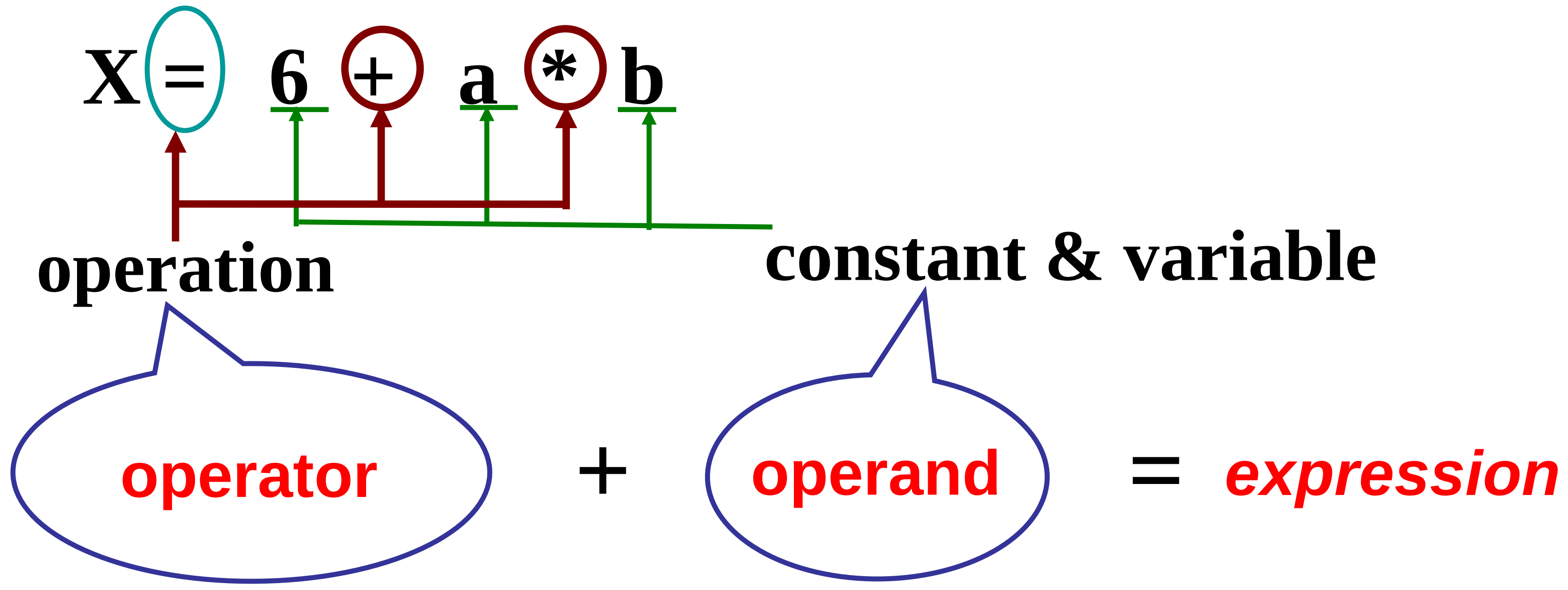
result: c

99

0 &'0' ?

Operators and Expressions

- Expression



Operators and Expressions

- Operator
  - **relationship to operands** $\left\{\begin{array}{l}\textbf{type}\\ \textbf{Operands}\left\{\begin{array}{l}\textbf{unary}\\ \textbf{binary}\\ \textbf{ternary}\end{array}\right.\\ \textbf{num}\end{array}\right.$
- Precedence: the order in which operator are applied.
- Associativity: within a level what’s the order of execution
  - left to right, mark as : $\rightarrow$
  - right to left, mark as: $\leftarrow$

Operators	
Arithmetic Operators	+,-,*,/,%
Relational Operators	>,<,>=,<=,==,!=
Logical Operators	&&,||,!
Bitwise Operators	&,|,^,>>,<<
Assignment Operators	= …
Increment and Decrement Operators	++,--
Conditional Operators	? :
Pointer Operators	&,*,->
Others……	( ), [ ],.,…

Operators and Expressions

- Precedence (P) & Associativity (A)

P	operator	A	P	operator	A
15	( ) [ ]	→	8	&	→
	. ->		7	^	→
14	++ -- ! ~ + - * & (type) sizeof	←	6	|	→
			5	&&	→
13	* / %	→	4	||	→
12	+ -	→	3	? :	→
11	<< >>	→	2	= += -= *= /= &= ^= |= <<= >>=	←
10	< <= >= >	→			
9	== !=	→	1	,	→

Arithmetic Operators and Expressions

- Arithmetic Operators

binary $\left\{ \begin{array}{ll} + & \text{addition} \\ - & \text{subtraction} \\ * & \text{multiplication} \\ / & \text{division} \\ \% & \text{modulus} \end{array} \right.$

unary $\left\{ \begin{array}{ll} + & \text{positive (omitted)} \\ - & \text{negative} \end{array} \right.$

- Precedence

+ positive (14)
- negative

→

*** multi**
/ division (13)
% modulus

→

+ add (12)
- subtract

Arithmetic Operators and Expressions

- Associativity : *from left to right ->*
  - % operator can be only applied to integers
  - if / is applied to two integers, the result will be an integer .

```
void main( )
{
     int a=5,b=7,c=-10,d;
     d= a% b +c;
    printf("%d\n",d);
     d= a% (b +c);
    printf("%d\n",d);
}
```

-5

2

```
void main( )
 { int a=6,b=8,c=9,d;
    d = a / b * c;
    printf("%d\n",d);
    d = 6.0/8*9;
     printf("%d\n",d);
}
```

0

6

Arithmetic Operators and Expressions

$$\frac{a+b+c}{\sqrt{a}+b(\sin x+\sin y+\sin z)}$$

$$\Downarrow$$

(a+b+c)/(sqrt(a)+b*(sin(x)+sin(y)+sin(z)))

$$\searrow \quad \downarrow \quad \downarrow \quad \swarrow$$

Mathematic functions

Relational Operators and Expressions

- Relational operators

  >, <, >=, <=, ==, != **(binary)**

true: 1
false: 0

- The result of relational expression is the logical value
- Precedence : **>, <, >=, <= (10) ==, !=** (9)
- Associativity : ***from left to right ->***

examples:

- **5>7>0>2<10** ⇒ **1**
- **1/3*3 == 1** ⇒ **0**
- **a = 25;**
- **x=100>a>10 ; x=?** ⇒ **0**

Relational Operators and Expressions

- Logical operators

  &&, || (binary), ! (unary)

true: 1

false: 0

- The result of logical expression is the logical value
- Precedence : *! (14), &&(5), || (4)*
- Associativity : **&&,||** ***from left to right -->, !*** **←**

Truth table for logical operators

a	b	!a	!b	a&&b	a||b
1	1	0	0	1	1
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	0	0

Relational Operators and Expressions

a=1, b=2,c=0,d=4, m=1,n=1

(c || b) && (c || a) $\Longrightarrow$ 1

c || b && c || a $\Longrightarrow$ 1

a < b && !a $\Longrightarrow$ 0

- a&&b&&c if a is non-zero， then test b, both a and b are non-zero， then test c
- a||b||c， if a is non-zero， needn’t test b and c, the result is 1.

(m=a>b)&&(n=c>d) $\Longrightarrow$ *m=0 n=1*

m=? n=?

Short-circuit evaluation

Relational Operators and Expressions

Question:

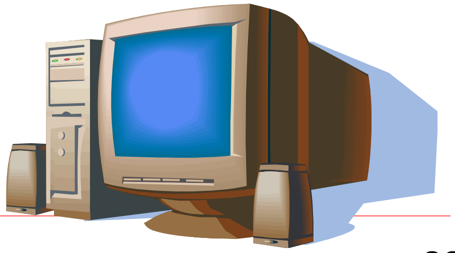
in mathematical form , if x 's value is between 0 and 10, we can write it like (0≤x≤10), In C, How to code it?

0 <=x <=10 **right? Or not?**

✗

If x = -1, result? **Forever 1**

$\boxed{0 <= x \ \&\& \ x <= 10}$



Bitwise Operators and Expressions

- Bitwise operators

  &, |,>>,<<,^ (binary), ~ (unary)

- Precedence : ~ (14), &(8), |(6), ^(7),<<, >>(11),

- Associativity : **&,|,^,<<,>>, *from left to right -->,* ~ ←**

Truth table for bitwise operators

a	b	a^b	~a	~b
1	1	0	0	0
1	0	1	0	1
0	1	1	1	0
0	0	0	1	1

Bitwise Operators and Expressions

int x=7, y=10; high bits are omitted here.

x & y=2 (0000 0111 **&** 0000 1010= 0000 0010)

x | y=15 (0000 0111 **|** 0000 1010= 0000 1111)

x ^ y=13 (0000 0111 **^** 0000 1010= 0000 1101)

~ x = -8 (**~** 0000 0111 = 1111 1000)

x<<2=28 (0000 0111 **<<**2 = 0001 1100)

y>>2= 2 (0000 1010 **>>**2 = 0000 0010)

Bitwise Operators and Expressions

Notes:

- x << 1 ⇔ x*2
- x >> 1 ⇔ x/2
- a & = b ⇔ a = a & b
- a << =2 ⇔ a = a << 2
- Bitwise operands can only be int, char or long type

Bitwise Operators and Expressions

```
#include <stdio.h>
void main()
  {  unsigned a,b,c,d;
      scanf("%o",&a);
      b=a>>4;
      c=~(~0<<4);
     d=b&c;
     printf("%o,%d\n%o,%d\n",a,a,d,d);
```

```
11011001
00001111
00001101
```

```
3 3 1      (Input)
3 3 1,  217  ( a )
1 5,  13     ( d )
```

Assignment Operators and Expressions

- Assignment operator **= (binary)**
- Result is the value assigned

variable = expression;
Where :
variable is the variable you wish to set,
can only hold a value of the appropriate type.
expression specifies the value

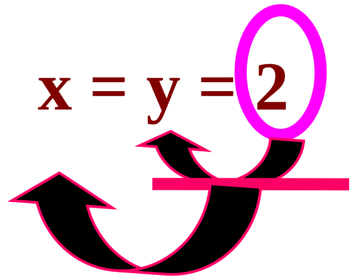
```
int x ;  float y;
y =10.2;
x = y +2;
```

→

```
y =10.2;
x = 12;
```

Assignment Operators and Expressions

- Precedence : = (2)
- Associativity : ***from right to left*** ←



- Compound assignment operators :

variable op= expression;

⇔ ***variable = variable op (expression);***

Where op can be:

+ – * / % << >> | & ^

Assignment Operators and Expressions

```
int x, a=5; x=a+=a-=a*=a;    x=?, a=?
```

0,0

```
int  a=5,b=10; a+=a+b;   a=?  b=?
```

20,10

```
int a=2,b=5,c=6,d=10;
a+=b ; b-=c ; c*=d ; d/=a;   a%=c;
a=? b=? c=? d=?
```

7,-1,60,1

```
int  x=4,y=5;
x += 2;    x *= y+1;
x >>=2;
x = ?  y = ?
```

9,5

Increment and Decrement

- Increment and Decrement operators

  ++,-- (unary)

- Result is the variable been added/subscribed 1

  ++x (Prefix) increments x before its value is used

  → **x =x+1**

  x++(Postfix) increments x after its value has been used

  --x(Prefix) decrements x before its value is used

  → **x =x-1**

  x--(Postfix) decrements x after its value has been used

Notes: **"++,--" only be used to int、char、long & pointers**

Precedence : ++,-- (14)

Increment and Decrement

```
void main( )
{
    int  a , b , s  ;
    a = 5 ; b = 5 ;
    s = a+b ;     printf(“%d,%d,%d\n”, a,b,s);
    s = a++ +b; printf(“%d,%d,%d\n”,a,b,s);
   s = ++a+b; printf(“%d,%d,%d\n”,a,b,s);
   s = --a +b; printf(“%d,%d,%d\n”,a,b,s);
   s = a-- +b; printf(“%d,%d,%d\n”,a,b,s);
   s = a + b;   printf(“%d,%d,%d\n”,a,b,s);
}
```

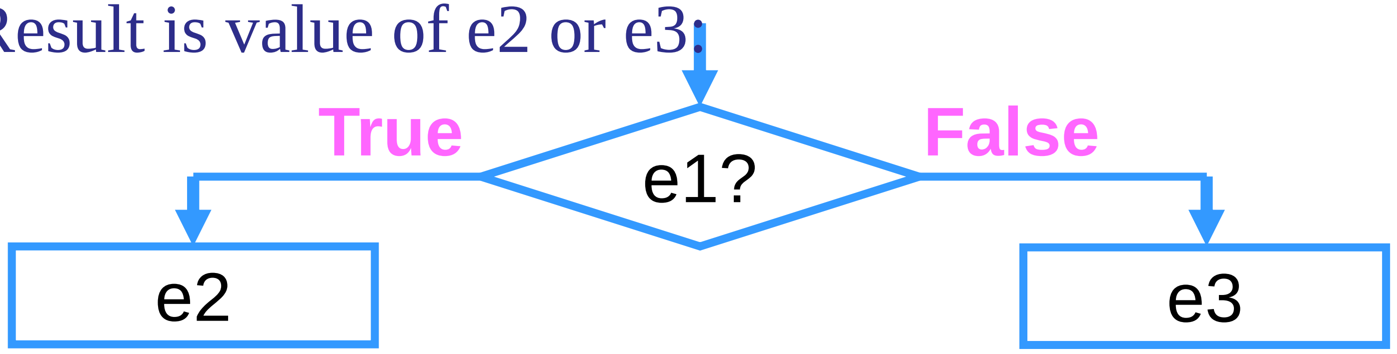
```
5,5,10
6,5,10
7,5,12
6,5,11
5,5,11
5,5,10
```

Conditional Operators and Expressions

- Conditional operators:

  ? : (ternary) , form is x = e1 ? e2 : e3 ;

- Result is value of e2 or e3:



- Precedence : *? : (3)*

Examples:

int x=2,y=3; a = (x>y) ? x+y : x-y **-1**

max = (x>y) ? x :y; min = (x<y) ? x : y; **3,2**

Comma Operators and Expressions

- Comma operator: $\underbrace{(\ \text{exp1,exp2,exp3…..,expn})}_{\text{comma expression}}$

- The result of the whole comma expression is the value of expn.

Examples:

$X = (\ (\ \boxed{a=4*5}\ , a*2\ )\ ,\ \underline{a+6}\ )$

$\underline{a=?}$ 20

$\underline{x=?}$ 26

Type Conversion

- *Question*

1+2.3 -> integer combined with floating point

int x; float y;

y = 1+2.3; x = y; -> assign floating point to integer variable.

automatic type conversion

Values of one type are converted into another compatible type as an implicit part of computation process.

Examples:

1+2.3 → 1 is converted into floating point number 1.0, then do addition.

x = y; → in assignment, the left value is converted into the type of right variable.

Type Conversion

- automatic type conversion rules:

$$\begin{tikzcd}
\textbf{Double} & \textbf{float} \arrow[l] \\
\textbf{long} \arrow[u, bend left] & \\
\textbf{unsigned} \arrow[u, bend left] & \\
\textbf{int} \arrow[u, bend left] & \textbf{char} \arrow[l]
\end{tikzcd}$$

explicit type conversion

Using type cast -> (data type) to do type conversion.

Examples:

```
int x =2, y = 5; float z;
z  = x/y;
z = double(x)/y;
```

$\Longrightarrow$

```
0.000000
0.400000
```

Summary

- Basic Data Types in C
  - Int，char，float，double
  - string
- Declaration & Initialization of Varaibles
- Operators and Expressions
  - Arithmetic operators and expressions
  - Relational operators and expressions
  - Logical operators and expressions
  - Bitwise operators and expressions
  - Assignment operators and expressions
- Type Conversion

TERMS

- Type conversion 类型转换
- Data type 数据类型
- Precedence 优先级
- associative 结合性
- Expression 表达式
- Operator 运算符
- Operand 操作数
- Modulo 求余数
- Division 除法
- Addition 加法
- Subtraction 减法
- Multiplication 乘法
- Prefix 前缀
- Postfix 后缀

TERMS

- Unary operator 单目运算符
- Binary operator 双目运算符
- Ternary operator 三目运算符
- Arithmetic operator 算术运算符
- Relational operator 关系运算符
- Logical operator 逻辑运算符
- Bitwise operator 位运算符
- Conditional operator 条件运算符
- Assignment operator 赋值运算符
- Increment operator 自增运算符
- Decrement operator 自减运算符

TERMS

- Explicit conversion 显式转换
- Implicit conversion隐式转换
- Assignment 赋值
- Symbolic constant 符号常量
- Qualifier 修饰符
- Address 地址
- Decimal 十进制的
- Octal 八进制的
- Hexadecimal 十六进制的
- Binary 二进制的

Thank you!