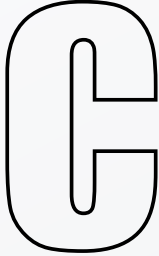


C Programming

huiw@suda.edu.cn

 **Programming**

程序设计语言

语言是句子的集合。

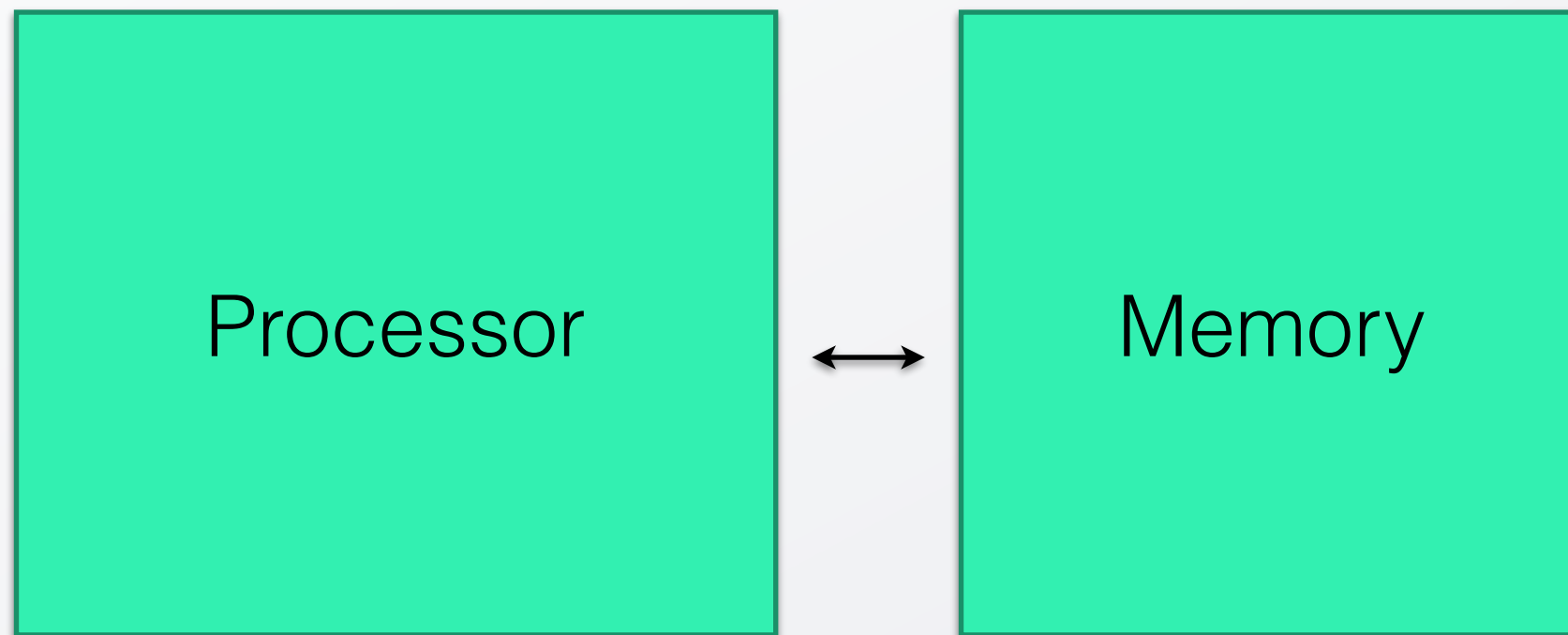
C语言是符合C语言语法的句子的集合。

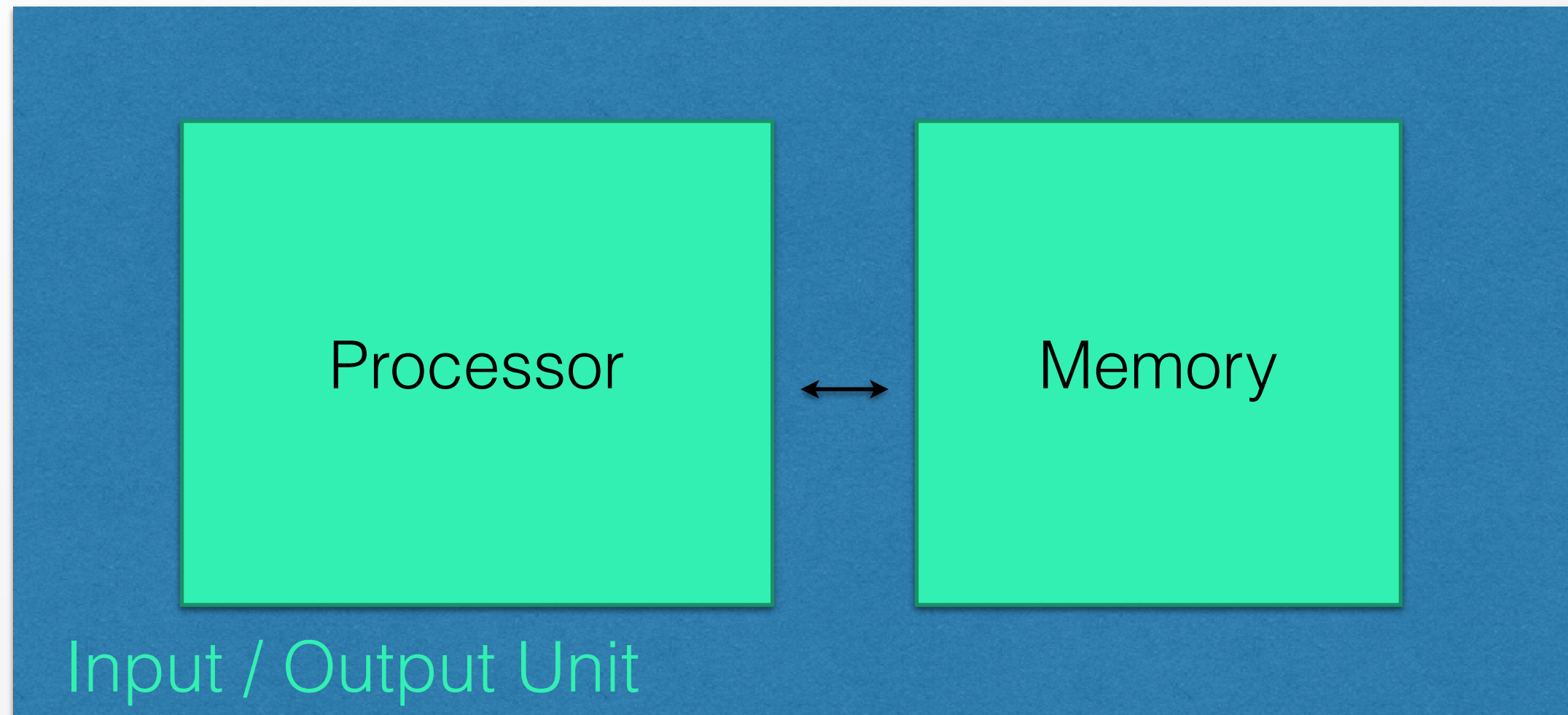
C语言中“句子”

可以命令计算机执行一个或者一组机器动作。

也可以定义或者描述某些数据对象。

所以，学习C语言首先需要
了解计算机结构和计算机可以
执行的基础动作。



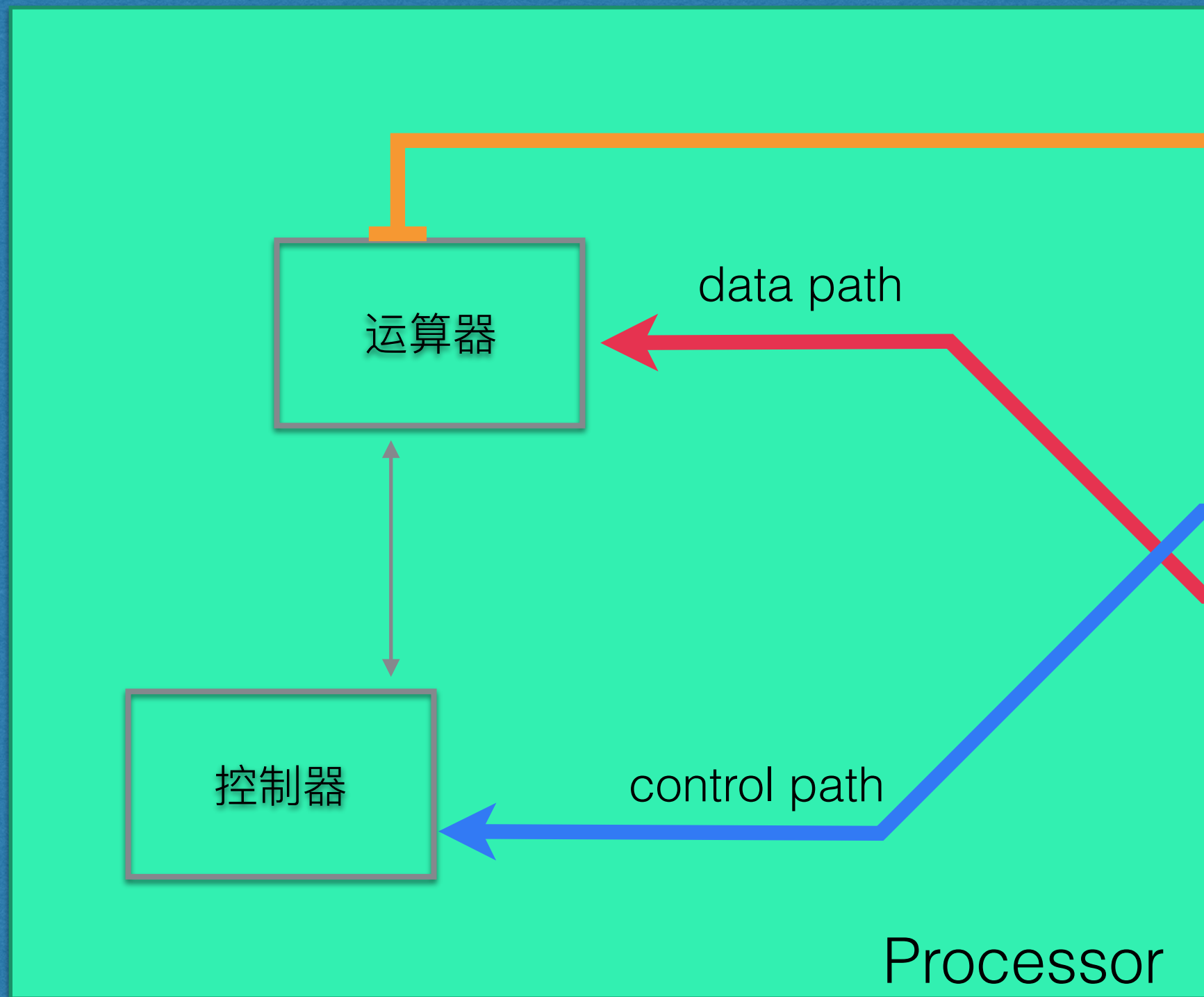




Processor

The diagram illustrates a computer system architecture. It features a large, light blue rectangular box on the left labeled "Processor". To its right is a smaller, light blue rectangular box labeled "Input / Output Unit". A horizontal black double-headed arrow connects the right side of the "Processor" box to the left side of the "Input / Output Unit" box, indicating bidirectional communication between the two components.

Input / Output Unit



计算机硬件层次的操作低级繁杂

C语言语句可以高效自然地
实现这些机器动作。

存储器操作

1

- 存与取



赋值语句*

```
int number_of_students;  
number_of_students = 60;
```

* 应该称为赋值表达式语句。为了方便大家记忆，简称为赋值语句。

赋值语句*

```
int number_of_students;  
number_of_students = 60;
```

number_of_students

? ? ?

赋值语句*

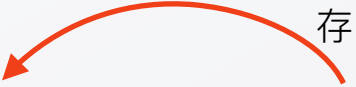
```
int number_of_students;  
number_of_students = 60;
```

number_of_students

60

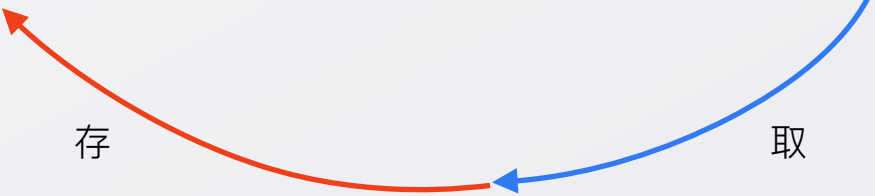
赋值语句*

```
int number_of_students;  
number_of_students = 60;
```



A red curved arrow points from the value 60 in the assignment statement to the variable name number_of_students in the declaration. The character '存' (store) is written above the arrow.

```
int another_variable_keeps_the_number;  
another_variable_keeps_the_number = number_of_students;
```



A blue curved arrow points from the variable name number_of_students to the assignment operator, and a red curved arrow points from the assignment operator to the variable name another_variable_keeps_the_number. The character '取' (retrieve) is written above the blue arrow, and the character '存' (store) is written below the red arrow.

赋值语句*

number_of_students

60

another_variable_keeps_the_number

???

赋值语句*

number_of_students

60

60

another_variable_keeps_the_number

???

赋值语句*

number_of_students

60

another_variable_keeps_the_number

60

赋值语句

```
int number_of_students;  
number_of_students = 60;
```

```
number_of_students = number_of_students;
```



number_of_students 变量在存储的位置



number_of_students 变量里面存储的内容

赋值语句

```
int number_of_students;  
number_of_students = 60;
```

number_of_students = number_of_students;



使用存储器的时候，我们给数据对象（整数变量）起个可唯一标识的名字，该名字串称为**标识符**。

注意：标识符的前31个字符为有效字符。

标识符的命名规则

D [0-9]

L [a-zA-Z_]

$\{L\} (\{L\} | \{D\})^*$

标识符的命名规则

- 全部使用小写字母
- 使用 '_' 作为单词间的分割符号
- 名字中可以使用单位

```
uint32 timeout_msecs;  
uint32 my_weight_lbs;
```

- 增加合适的前缀和后缀

标识符举例

ALLUPPERCASENAME

is_prime

a
j, k, l

data

aa
jj, kk, ll

search_counter

xueshengrenshu

标识符举例

~~ALLUPPERCASENAME~~

is_prime

~~a~~

~~j, k, l~~

~~data~~

~~aa~~

~~jj, kk, ll~~

search_counter

~~xueshengrenshu~~

标识符举例

~~ALLUPPERCASENAME~~

is_prime

a
j, k, l

~~data~~

aa
~~jj, kk, ll~~

search_counter

~~xueshengrenshu~~

避免使用容易混淆的字符

s, 5

9, g

1, i, l

0, o, O

z, 2

运算器操作

- 算术运算
- 逻辑运算
- 关系运算



算术运算

$+$ $-$ $*$ $/$ $\%$

加 减 乘 除 余

注意： C语言中的%运算是取余运算， 而非取模运算。

$$a^2 - 2ab + b^2$$

```
int a;  
int b;
```

```
int result;
```

```
result = a*a + 2*a*b + b*b;
```



```
int a1;  
int b1;
```

```
int a2;  
int b2;
```

```
int result;
```

```
result = a1*b2 - a2*b1;
```

$$\begin{vmatrix} \mathbf{a}_1 & \mathbf{b}_1 \\ \mathbf{a}_2 & \mathbf{b}_2 \end{vmatrix}$$

算术运算

++

加 1

--

减 1

逻辑运算

! && ||

非 与 或

关系运算

$==$ $!=$ $<$ $<=$ $>$ $>=$

相等 不等 小于

大于

小于等于

大于等于

?

输入 a, b, c 三条线段的长度，判断这三条线段是否能围成一个三角形。

Precedence	Name	Symbol(s)	Associativity
1	increment (postfix)	++	left
	decrement (postfix)	--	
2	increment (prefix)	++	right
	decrement (prefix)	--	
	unary plus	+	
	unary minus	-	
3	multiplicative	* / %	left
4	additive	+ -	left
5	assignment	*= /= %= += -=	right

控制器操作

- 条件语句

- 循环语句

- 直接跳转语句



选择语句 (if...else...)

```
selection_statement  
    : if '(' expression ')' statement  
    | if '(' expression ')' statement else  
    statement
```



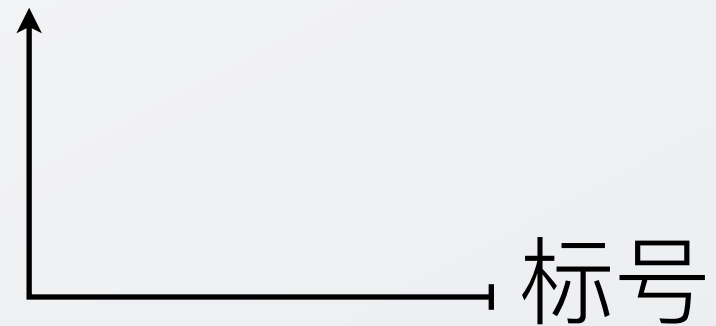
```
int max(int a, int b) {  
    if (a < b)  
        return b;  
  
    return a;  
}
```

```
int max(int a, int b) {  
    if (a < b)  
        return b;  
    else  
        return a;  
}
```

跳转语句 (goto)

jump_statement

: **goto** IDENTIFIER ';' ;



```
unsigned int  
sum_of_squares(unsigned int n)  
{  
    unsigned int sum = 0;  
    unsigned int current_number = 1;
```

→ loop:

```
    sum = sum + (current_number * current_number);
```

```
    current_number++;
```

```
    if (current_number <= n) .
```

→ goto loop;

```
    return sum;
```

```
}
```

$$\sum_{i=0}^n i^2 = \frac{1}{6}n(n+1)(2n+1)$$

```
unsigned int  
sum_of_squares_fast(unsigned int n)  
{  
    return n * (n+1) * (2*n + 1) / 6;  
}
```

抽象机制

操作抽象与数据抽象

函数


```
int  
square(int x)  
{  
    return x * x;  
}
```

```
bool  
is_odd(int i)  
{  
    return (i % 2 == 1);  
}
```

```
bool  
is_even(int i)  
{  
    return !is_odd(i);  
}
```

数据抽象

将数据的使用 与
数据的表示和实现技术分离。

小结

抽象数据类型

函数 与 复合数据类型

存储操作

运算符与表达式

程序流程控制

抽象数据类型

函数 与 复合数据类型

func()

int
unsigned int

存储操作

=

运算符与表达式

+ - * / % ! || && == != < > <= >=

程序流程控制

goto if...else...

程序调试

C库函数

C语言预处理

附录

大端和小端

例如： 0x90AB12CD

Big Endian

1000	1001	1002	1003
90	AB	12	CD

Little Endian

1000	1001	1002	1003
CD	12	AB	90