中山大學
SUN YAT-SEN UNIVERSITY

选择结构

中山大学计算机学院

讲课人：张晓溪

目录

1.布尔表达式
2.选择语句(if-else)
    1.单路选择
    2.双路选择
    3.多路选择
3.嵌套if-else语句
4.switch语句
5.三元条件运算符

语句(Statements)

Statements	Explanation	Syntax	Example

复合语句
(Compound Statements)
	复合语句又称为块(Block)，是由花括号{}所包围的语句与声明的序列。	{
		expression; or
		declaration; or
		statement
		}
			int main() {
			int n = 1;
			n = n * n;
			printf("n*n = %d\n", n);
			return 0;
			}

表达式语句 (Expression Statements)	由分号;结尾的表达式。C语言中大多数语句都是表达式语句。	expression;	n = 1; or n = n * n; or printf("n*n = %d\n", n);

选择语句
(Selection Statements)
	选择语句根据表达式的值，选择数条语句中的一条来执行。	if (expression)
		statement
		else statement
			if (n > 0) {
			printf("n > 0\n");
			} else {
			printf("n < 0\n");
			}

循环语句
(Iteration Statements)
	循环语句重复执行一条语句。	while (expression)
		statement
		for (init-exp;exp;exp)
		statement
			for (int i = 0; i < 10; i++) {
			int i2 = i*i;
			printf("%d^2 = %d\n",i,i2);
			}

跳转语句 (Jump Statements)	跳转语句无条件地转移程序控制流。	break; continue; return expression; goto label;	return 0;

布尔表达式(Boolean Expression)

布尔表达式是一个计算结果为真(True)或假(False)的表达式。在编程实践中，布尔表达式通常用于判断语句、循环语句等其他需要条件判断的场合。

在C语言中，布尔表达式包括：

- 比较表达式，例如x == y、 count <= 5
- 逻辑表达式，例如 (x == y) && (count <= 5)、!(y >= 3)
- 使用(_Bool)强制转换为布尔类型(非0则转为1)

C语言对布尔类型的支持

C语言内置支持：

- 布尔常数：1，0
- 布尔类型：_Bool

通过#include <stdbool.h>
宏扩展：

- #define true 1
- #define false 0
- #define bool _Bool

```c
/* bool.c */
#include <stdio.h>
#include <stdbool.h>

int main() {
    _Bool x = 7;
    bool y = 15;
    printf("      _Bool x: %x\n", x);
    printf("       bool y: %x\n", y);
    printf(" sizeof(true): %lu\n", sizeof(true));
    printf("sizeof(false): %lu\n", sizeof(false));
    printf("    sizeof(x): %lu\n", sizeof(x));
    printf("    sizeof(y): %lu\n", sizeof(y));
    printf("sizeof(_Bool): %lu\n", sizeof(_Bool));
    printf(" sizeof(bool): %lu\n", sizeof(bool));
    return 0;
}
```

C语言对布尔类型的支持

通过#include <stdbool.h>宏扩展：

- #define true 1
- #define false 0
- #define bool _Bool

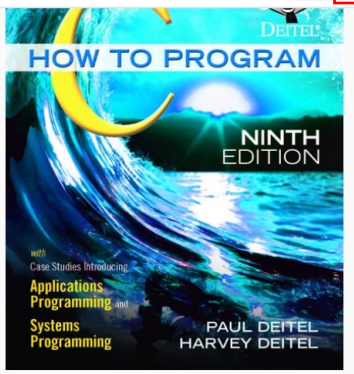
true被替换为1
false被替换为0

输出：

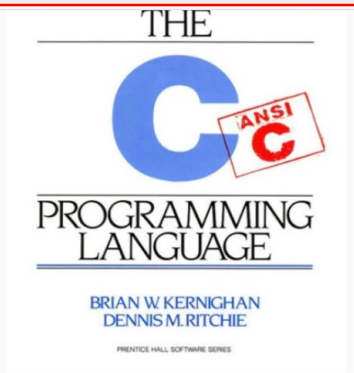
```
      _Bool x: 1
       bool y: 1
 sizeof(true): 4
sizeof(false): 4
    sizeof(x): 1
    sizeof(y): 1
sizeof(_Bool): 1
 sizeof(bool): 1
```

_Bool类型只占用1个字节

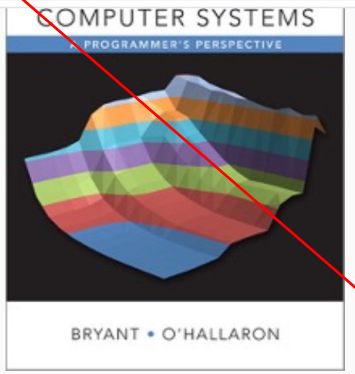
```c
/* bool.c */
#include <stdio.h>
#include <stdbool.h>

int main() {
    _Bool x = 7;
    bool y = 15;
    printf("      _Bool x: %x\n", x);
    printf("       bool y: %x\n", y);
    printf(" sizeof(true): %lu\n", sizeof(true));
    printf("sizeof(false): %lu\n", sizeof(false));
    printf("    sizeof(x): %lu\n", sizeof(x));
    printf("    sizeof(y): %lu\n", sizeof(y));
    printf("sizeof(_Bool): %lu\n", sizeof(_Bool));
    printf(" sizeof(bool): %lu\n", sizeof(bool));
    return 0;
}
```



C How to Program (9th Edition)



The C Programming Language (2nd Edition)



Computer Systems: A Programmer's Perspective (3rd Edition)

Date	Lecture	PPT格式	PDF格式	Source
SEP08	第一讲 C语言概述	ppt	pdf	hello.c, printf_examples.c Matrix系统使用手册（Matrix.pdf）
SEP11	第二讲 变量常量	ppt	pdf	data_types.c, declaration.c, lvalue.c, char_val.c, const_define.c, format_str.c, storage_class.c, type_cast.c
SEP15	第二讲 变量常量与二进制	ppt	pdf	dec2bin.c, negative.c, min_max.c, min_max_ll.c, overflow_impl.c, overflow.c, overflow2.c, overflow3.c, var_addr_value.c, endian.c
SEP18	第三讲 运算符	ppt	pdf	arithmetic.c, type_conv.c, bitwise.c, relational.c, logical.c, assign.c, assign2.c, ternary.c, ternary2.c, sizeof.c, comma.c
SEP22	第四讲 选择结构	ppt	pdf	bool.c, ~~bool2.c, if0.c, if1.c, if2.c, if3.c, if4.c~~, score.c, sort3.c, ternary.c, weekday.c
SEP25	第五讲 循环结构			
SEP29	第五讲 控制结构进阶与练习			
OCT09	第六讲 函数与作用域			
	讲 函数进阶			

课件

点击源代码链接

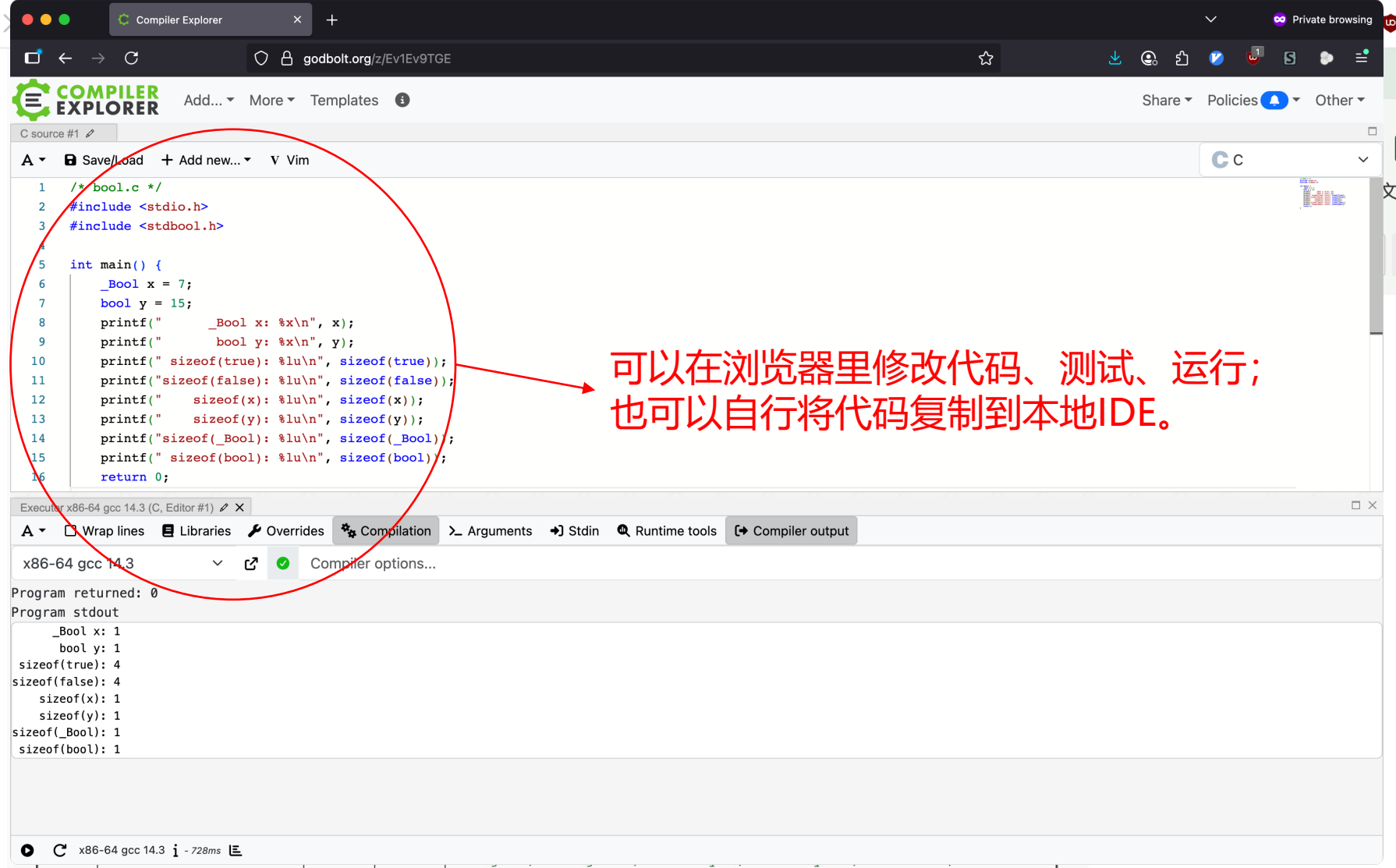
登陆课程主页

快捷添加

测验 作业 讨论区 文件库 考勤

课程简介 添加

https://godbolt.org/z/Ev1Ev9TGE



可以在浏览器里修改代码、测试、运行；
也可以自行将代码复制到本地IDE。

SEP22	第四讲 选择结构	ppt	pdf	bool.c, ~~bool2.c, if0.c, if1.c, if2.c, if3.c, if4.c~~, score.c, sort3.c, ternary.c, weekday.c
SEP25	第五讲 循环结构			
SEP29	第五讲 控制结构进阶与练习			
OCT09	第六讲 函数与作用域			
	讲 函数进阶			

课件

点击源代码链接

布尔表达式案例

案例要点：

- 生成一个0到9的随机数
  - srand(time(NULL))设置种子
  - rand()随机生成一个0 ~ RAND_MAX之间的整数
  - rand()%10的随机范围为0到9
- 判断n = rand()%10是否在(2,8]这个区间

```c
/* bool2.c */
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <time.h>

int main() {
    // Seed the random number generator
    srand(time(NULL));
    // Generate a random number between {0,n-1}
    int n = rand() % 10;

    printf("%d, %u\n", n, n > 2 && n <= 8);
    printf("%d, %u\n", n, n <= 2 || n > 8);
    printf("%d, %u\n", n, !(n > 2 && n <= 8));
    return 0;
}
```

条件与路径选择

在编程实践中，除了按照顺序执行以外，我们还经常需要根据某个条件来判断程序执行的路径。例如，我们需要通过判断学生的分数来决定输出其对应的评级：

```
IF 学生分数大于或等于90分:
    PRINT "A"
ELSE IF 学生分数大于等于80分:
    PRINT "B"
...
```

在C语言中，这样的路径选择是通过选择语句来实现的，其中又分为**单路选择**、**双路选择**、**多路选择**。

选择语句 - 单路选择

单路选择是指在某种条件下执行特定语句，而在其他条件下不执行任何代码的结构。其形式通常是：

```
if (expression)
    statement;
```

- if语句中表达式(expression)是布尔表达式
- 表达式必须在括号内
- 如果表达式不是常规布尔表达式，则会发生强制转换，非0值表示真，0表示假
- statement可以是复合语句

```c
/* if0.c */
#include <stdio.h>

int main() {
    int grade;
    scanf("%d", &grade);
    if (grade >= 90)
        printf("A\n");
    return 0;
}
```

选择语句 - 单路选择

单路选择是指在某种条件下执行特定语句，而在其他条件下不执行任何代码的结构。其形式通常是：

```
if (expression) {
    statement;
    statement;
    ...
}
```

如果执行多个语句的话，需要用花括号{}包围起来。

```c
/* if1.c */
#include <stdio.h>

int main() {
    int grade;
    scanf("%d", &grade);
    if (grade >= 90) {
        printf("A\n");
        printf("Congratulations!\n");
    }
    return 0;
}
```

选择语句 - 双路选择

双路选择是指在某种条件下执行特定语句，而在**其他条件下执行另一组特定语句**的结构。其形式通常是：

```
if (expression) {
    statement;
} else {
    statement;
}
```

```c
/* if2.c */
#include <stdio.h>

int main() {
    int grade;
    scanf("%d", &grade);
    if (grade >= 60) {
        printf("Passed\n");
    } else {
        printf("Failed\n");
    }
    return 0;
}
```

选择语句 - 多路选择

多路选择是有多个不同的判断条件，根据不同的判断条件执行其对应的特定语句。其形式通常是：

```
if (expression1) {
    statement;
} else if (expression2) {
    statement;
} else if (expression3) {
    statement;
} ...
  else {
    statement;
}
```

```c
/* if3.c */
#include <stdio.h>

int main() {
    int grade;
    scanf("%d", &grade);
    if (grade >= 90) {
        printf("A\n");
    } else if (grade >= 80) {
        printf("B\n");
    } else if (grade >= 70) {
        printf("C\n");
    } else if (grade >= 60) {
        printf("D\n");
    } else {
        printf("Failed\n");
    }
    return 0;
}
```

嵌套选择语句(Nested Selection Statement)

if...else语句中也可以包含一个或多个另外的if...else语句，称为**嵌套选择语句**。例如：

```
if (expression1) {
    if (expression2)
        statement;
    else if (expression3)
        statement;
    else
        statement;
} else {
    statement;
}
```

```c
/* if4.c */
#include <stdio.h>

int main() {
    int grade;
    scanf("%d", &grade);
    if (grade >= 90) {
        printf("A\n");
    } else {
        if (grade >= 80) {
            printf("B\n");
        } else {
            if (grade >= 70) {
                printf("C\n");
            } else {
                if (grade >= 60) {
                    printf("D\n");
                } else {
                    printf("Failed\n");
                }
            }
        }
    }
    return 0;
}
```

嵌套选择语句(Nested Selection Statement)

嵌套选择语句最好使用花括号{}来限定执行语句。
如果不使用{}，那么else总是与在其之前**未配对**的**最近**的if相匹配。

```
if (a != b)
if (a > b) printf("a > b\n");
else printf("a < b\n");
else printf("a == b\n");
```

```
if (a != b)
    if (a > b)
        printf("a > b\n");
    else
        printf("a < b\n");
else
    printf("a == b\n");
```

if(expr)后面的语句开始缩进4个空格
else匹配最近的未匹配的if，其后面的语句按照与else同级别的if进行缩进

嵌套选择语句(Nested Selection Statement)

嵌套选择语句最好使用花括号{}来限定执行语句。
如果不使用{}，那么else总是与在其之前**未配对**的**最近**的if相匹配。

```
int a = 1, b = 3, c = 5, d = 4, x;
if (a < b)
if (c<d)
x = 1;
else
if (a<c)
if (b<d)
x = 2;
else
x = 3;
else
x = 6;
else
x = 3;
```

```
int a = 1, b = 3, c = 5, d = 4, x;
if (a < b)
    if (c < d)
        x = 1;
    else
        if (a < c)
            if (b < d)
                x = 2;
            else
                x = 3;
        else
            x = 6;
else
    x = 3;
```

代码规范(Coding Style)

代码规范（推荐https://github.com/MaJerle/c-code-style）

- 不使用TAB，只用空格
- 缩进4个空格
- 在关键字与括号()之间留1个空格
- 左花括号{应与相应关键字处在同一行，即

```
if (x > 0) {
    printf("x > 0\n");
}
```

而非

```
if (x > 0)
{   printf("x > 0\n");
}
```

- 如果if-else某个分支使用了花括号的话，其他分支也必须使用大括号(即使该分支只有单个语句)
- ...

代码规范(Coding Style)

```
# include <stdio.h>
int n,m,v;
int main(){
  scanf("%d %d %d",&m,&n,&v);
  if (n>=m) {printf("-1 -1"); return 0;}
  if (v<=n) printf("%d %d",3,10);
  else if (v>=m) {
    if (v<=m*1.2) printf("%d %d",3,10*(v-m));
    else if (v<=m*1.5) printf("%d
%d",6,(int)(2*m+(v-1.2*m)*20));
    else printf("%d %d",12,(int)(8*m+50*(v-
1.5*m)));
  }
    else printf("0 0");
       return 0;
       }
```

```
#include <stdio.h>
int n, m, v;
int main() {
    scanf("%d %d %d", &m, &n, &v);
    if (n >= m) {
        printf("-1 -1");
        return 0;
    }
    if (v <= n)
        printf("%d %d", 3, 10);
    else if (v >= m) {
        if (v <= m * 1.2)
            printf("%d %d", 3, 10 * (v - m));
        else if (v <= m * 1.5)
            printf("%d %d", 6, 2*m + (v-1.2*m)*20);
        else
            printf("%d %d", 12, 8*m +50*(v-1.5*m));
    } else
        printf("0 0");
    return 0;
}
```

switch语句

```
switch (表达式) {
    case 常量1:
        statement_1; break;
    case 常量2:
        statement_2; break;
    case 常量3:
        statement_3; break;
    case 常量n:
        statement_n; break;
    default:
        statement; break;
}
```

- switch语句通常用来实现多路选择结构。switch语句的**可读性**比多分支if-else语句要更好
- switch语句括号中的"表达式"的值必须为**整数类型**
- 冒号:表示一个标签(Label)，不作为执行语句。
- case标签后面紧跟着一个**整型常量表达式**，即1，2，'A'，3+4等等，不可以是浮点数，不可以是变量(包括const常变量)
- 每一个case后面的常量值必须**各不相同**。

switch语句

```
switch (表达式) {
    case 常量1:
        statement_1; break;
    case 常量2:
        statement_2; break;
    case 常量3:
        statement_3; break;
    case 常量n:
        statement_n; break;
    default:
        statement; break;
}
```

- switch结构执行时，通常先计算括号()内表达式的值，然后将这个值逐个按照顺序与case标签中的常量进行比较。
- 如果表达式的值与某一个case标签的**常量n**相等，那么当前程序跳转到该case标签后面的语句，即statement_n，直至遇到break;跳出整个switch结构
- 如果表达式的值与所有的case常量都不相等，且该switch结构有一个default标签的话，则当前程序跳转到default:后面的语句

switch语句

```c
/* weekday.c */
#include <stdio.h>

int main() {
    int day;
    printf("Enter Day of Week [1-7]: ");
    scanf("%d", &day);
    switch (day) {
        case 1: printf("Monday\n"); break;
        case 2: printf("Tuesday\n"); break;
        case 3: printf("Wednesday\n"); break;
        case 4: printf("Thursday\n"); break;
        case 5: printf("Friday\n"); break;
        case 6: printf("Saturday\n"); break;
        case 7: printf("Sunday\n"); break;
        default: printf("Error\n"); break;
    }
    return 0;
}
```

输出：

```
$ ./weekday
Please Enter Day of Week [1-7]: 1
Monday
$ ./weekday
Please Enter Day of Week [1-7]: 3
Wednesday
$ ./weekday
Please Enter Day of Week [1-7]: 6
Saturday
$ ./weekday
Please Enter Day of Week [1-7]: 8
Error
```

switch语句

```c
/* weekday.c */
#include <stdio.h>

int main() {
    int day;
    printf("Enter Day of Week [1-7]: ");
    scanf("%d", &day);
    switch (day) {
        case 1: printf("Monday\n"); break;
        case 2: printf("Tuesday\n"); break;
        case 3: printf("Wednesday\n"); break;
        case 4: printf("Thursday\n"); break;
        case 5: printf("Friday\n"); break;
        case 6: printf("Saturday\n"); break;
        case 7: printf("Sunday\n"); break;
        default: printf("Error\n"); break;
    }
    return 0;
}
```

如果把break;删掉

输出：

```
$ ./weekday
Please Enter Day of Week [1-7]: 3
Wednesday
Thursday
Friday
Saturday
Sunday
Error
```

day == 3
程序跳转到case 3:以后顺序执行，由于没有遇到break;语句，所以会一直执行至switch语句末尾}

switch语句

```c
/* score.c */
#include <stdio.h>

int main() {
    int score;
    printf("Enter Score [0-100]: ");
    scanf("%d", &score);
    switch (score / 10) {
        case 10:
            printf("100!! Congratulations!\n");
        case 9: printf("A\n"); break;
        case 8: printf("B\n"); break;
        case 7: printf("C\n"); break;
        case 6: printf("D\n"); break;
        default: printf("F(ailed)\n");
    }
    return 0;
}
```

是否使用break;语句取决于实际情况

输出：

```
$ ./score
Enter Score [0-100]: 59
F(ailed)
$ ./score
Enter Score [0-100]: 80
B
$ ./score
Enter Score [0-100]: 98
A
$ ./score
Enter Score [0-100]: 100
100!! Congratulations!
A
```

switch语句

关于case后面的**整型常量表达式**：

```
const int a = 42;
switch (expr) {
    case 10: printf("..."); break;        //正确
    case 8+9: printf("..."); break;       //正确
    case 'A': printf("..."); break;       //正确，字符和整数可以相互转换
    case 'A'+19: printf("..."); break;    //正确，字符和整数可以相互转换
    case 9.5: printf("..."); break;       //错误，不能为小数
    case a: printf("..."); break;         //错误，不能包含变量
    case a+10: printf("..."); break;      //错误，不能包含变量
}
```

三元条件运算符

三元条件运算符? :用来组成一个条件表达式(Conditional Expression)。其基本语法为:

```
Expression1 ? Expression2 : Expression3
```

该表达式的值取决于Expression1的值(真或假)。如果Expression1为真，那么该表达式的值即Expression2的值；如果Expression1为假，那么该表达式的值即Expression3的值。相当于选择判断语句:

```
if (Expression1) {
    Expression2;
} else {
    Expression3;
}
```

三元条件运算符

```c
/* ternary.c */
#include <stdio.h>

int main() {
    int a = 33, b = 22, c = 44;;
    printf("  Max(a,b): %d\n", a > b ? a : b);
    printf("  Min(a,b): %d\n", a < b ? a : b);
    printf("Max(a,b,c): %d\n",
            a > b ?
            (a > c ? a : c) :
            (b > c ? b : c));
    return 0;
}
```

输出:

```
  Max(a,b): 33
  Min(a,b): 22
Max(a,b,c): 44
```

三元条件运算符

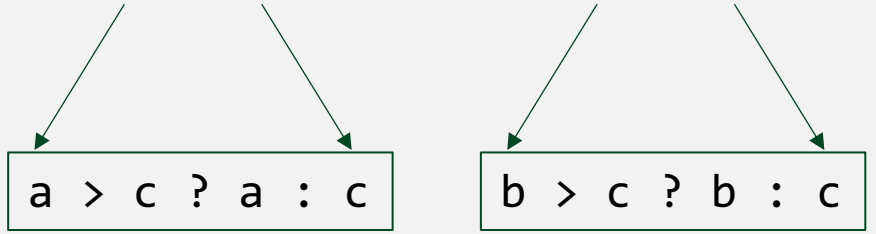
```c
/* ternary.c */
#include <stdio.h>

int main() {
    int a = 33, b = 22, c = 44;;
    printf("  Max(a,b): %d\n", a > b ? a : b);
    printf("  Min(a,b): %d\n", a < b ? a : b);
    printf("Max(a,b,c): %d\n",
            a > b ?
            (a > c ? a : c) :
            (b > c ? b : c));
    return 0;
}
```

输出:

```
  Max(a,b): 33
  Min(a,b): 22
Max(a,b,c): 44
```

嵌套(Nested Ternary Operator):

```
a > b ?    (exp2)      :     (exp3)
```



Sort 3 Numbers

```c
/* sort3.c */
#include <stdio.h>

int main() {
    int a, b, c;
    int max, min, median;
    scanf("%d%d%d", &a, &b, &c);

    max = (a > b ? a : b) > c ? (a > b ? a : b) : c;
    min = (a < b ? a : b) < c ? (a < b ? a : b) : c;
    median = a + b + c - max - min;

    printf("%d %d %d\n", min, median, max);
    return 0;
}
```

输出:

```
$ ./sort3
3 5 2
2 3 5
```

```c
max_ab = a > b ? a : b;
max = max_ab > c ? max_ab : c;

min_ab = a < b ? a : b;
min = min_ab < c ? min_ab : c;
```

FAQ

`a + 42` 是一个**表达式**
`a + 42;` 是一个**语句**(由分号结尾)

赋值表达式的值是被赋的值。例如`int x; x = 5;`
`x = 5`这个表达式的值(value)就是5
这个特性可以让我们进行链式赋值:

```
int a, b, c;
a = b = c = 10;   相当于   a = (b = (c = 10));
```

闰年的判断(leap year):

```
year % 400 == 0 || (year % 4 == 0 && year % 100)
```

FAQ

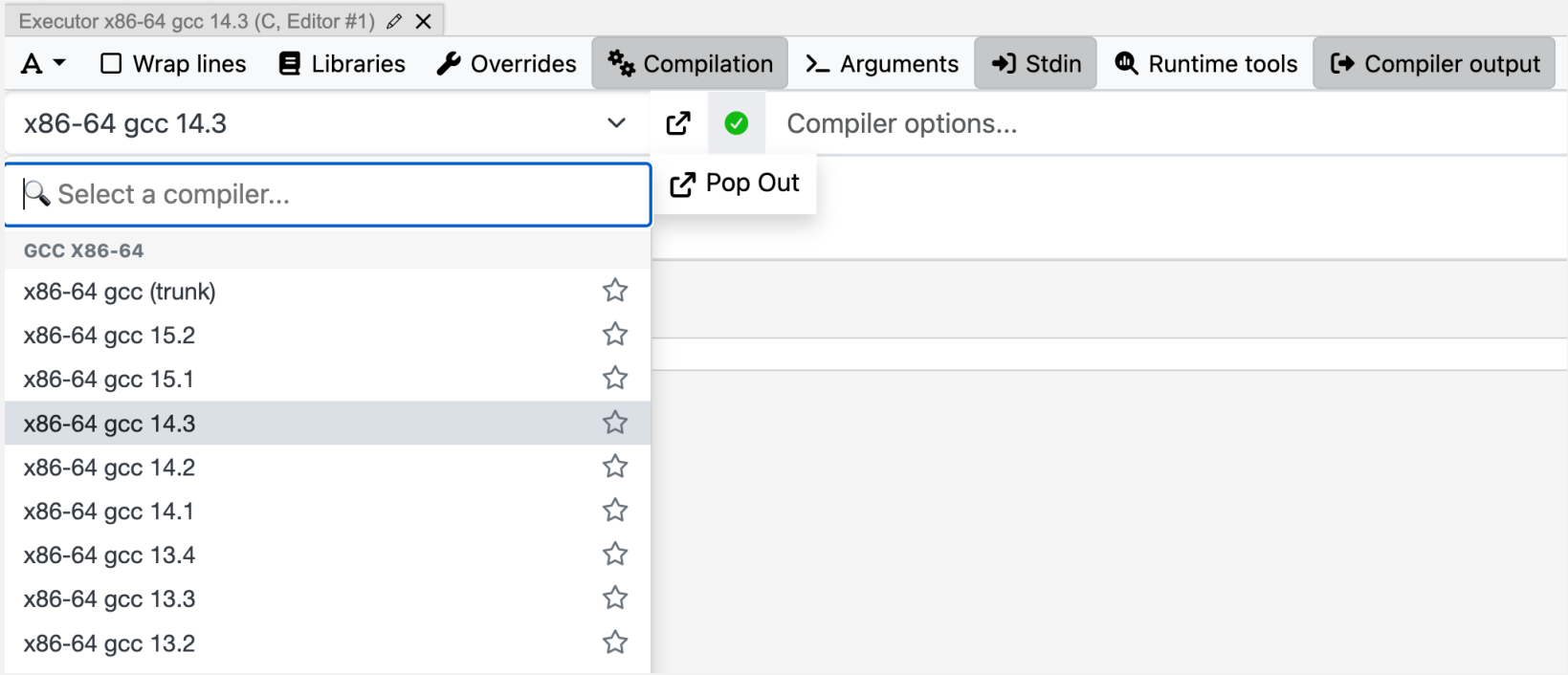
在Matrix提交时请使用C语言，不要使用C++

```cpp
/* C++ */
#include <iostream>
#include <cstdio>

using namespace std;

int main() {
    cout << "Hello, world" <<endl;
    return 0;
}
```

```c
/* C */
#include <stdio.h>

int main() {
    printf("Hello, world\n");
    return 0;
}
```

FAQ

Matrix里运行C语言代码时使用的编译器是x86_64 gcc 14，同学们在godbolt.org(或其他在线编译器)请选择相应的编译器。
千万不要选gcc 15！(gcc15相对于gcc14是一次重大升级，特定情况下同样的代码在这两个不同版本下的输出可能会不一样。)



中山大學
SUN YAT-SEN UNIVERSITY

谢谢

中山大学计算机学院