بسم الله الذي لا يضر مع اسمه شيء نبدأ :-

اليوم موضوعنا عن أوامر التحكم وتركيبها .. متفقين بشكل عام ان تنفيذ البرنامج يتم بصورة متتالية لكن بعض الجمل قد احتاج لتنفيذها شرط في حال كان الشرط صحيح .. يتم تنفيذ الجملة وفي حال كان خاطئ يتم تجاهلها ..

النوع الاول من جمل التحكم ..جمل التحكم التي يتوقف تنفيذها على شرط معين ..

selection statement

Relational Operators

العمليات التي تقوم بفحص العلاقة بين متغيرين وعنصرين يكون ناتجها اما صحيح او خاطئ (true or false)

عملية المقارنة تحتاج لعنصرين عن يمين المعامل وعن شماله ..لفحصهم "وهذا يتوافق كلياً مع المنطق" ..في حال مقارنة اشطر واحد بالشعبة يتم مقارنة شخص وهو العنصر الاول ببقية الصف وهو العنصر الثاني .. لا يمكن للمقارنة ان تكون لعنصر واحد "يخالف المنطق" (الا اذا كان الدكتور شاطر كثير وفيش غير طالب واحد مسجل عندو ^^"هاي مزح مش حفظ")

الجدول التالي يوضح هذه المعاملات وسيتم شرح كل منها تفصيلياً

TABLE 4-1 Relational Operators in C++

Operator	Description
==	equal to
!=	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

المعامل الاول (==):-

يستخدم في حال السؤال هل العنصر الاول = العنصر الثاني .. مثال :-

6==5➔false

||10==10➔true||4==2➔false||10==5*2➔true||15==30/2➔true

||and so on..

المعامل الثاني (=!) :-

يعني لا يساوي وهو عكس المعامل الاول وكأننا نسال هل العنصر الاول لا يساوي العنصر الثاني ..مثال :-

5!=10➔true||5!=5➔false||4!=2*2➔false||15!=30/2➔false||and so on.....

المعامل الثالث (>):-

يستخدم في حال السؤال هل العنصر الاول اقل من العنصر الثاني ..مثال :-

5<3➔false||10<15➔true||50<30*2➔false||100<80*2➔false||100<100➔false||and so on.

المعامل الرابع (=>):-

يستخدم في حال السؤال هل العنصر الاول اقل او يساوي العنصر الثاني في حال كان العنصر الاول يساوي العنصر الثاني او اقل منو الناتج true (يعني تحقيق واحد من الشرطين يكفي) ..مثال :-

5<=5➔true||100<=200➔true||30<=60/2➔true||20<=20➔true||and so on…

المعامل الخامس (>):-

يستخدم في حال السؤال هل العنصر الاول اكبر من العنصر الثاني .. مثال :-

50>20➔true||30>50➔false||100>50*2➔false||90>90➔false||and so on….

المعامل السادس (=<):-

يستخدم في حال السؤال هل العنصر الاول اكبر او يساوي العنصر الثاني في حال كان العنصر الاول اكبر او يساوي العنصر الثاني الناتج true (يعني تحقيق واحد من الشرطين بكفي)..مثال :-

50>=50➔true||100>=50➔true||90>=100➔false||and so on….

يمكننا استخدام المعاملات ذات العلاقات (التي تقوم بفحص العلاقة بين عنصرين) في ثلاثة انواع من البيانات .

١-الصحيحة (البسيطة) "int" مثال

10>5➔true

٢- العشرية "float" مثال :-

1.0 == 3.0/7.0 + 2.0/7.0 + 2.0/7.0 evaluates to false

Why? 3.0/7.0 + 2.0/7.0 + 2.0/7.0 = 0.99999999999999989

**في المثال السابق يجب مراعاة الاولويات في العمليات الحسابية

٣-الرموز "char" :- هنا يعتمد على الاسكي كود الخاص بكل رمز .. حيث يتم مقارنة قيمة الاسكي كود بين العنصرين مثال

TABLE 4-2 Evaluating Expressions Using Relational Operators and the ASCII Collating Sequence

Expression	Value of Expression	Explanation
`' ' < 'a'`	`true`	The ASCII value of `' '` is 32, and the ASCII value of `'a'` is 97. Because `32 < 97` is `true`, it follows that `' ' < 'a'` is `true`.
`'R' > 'T'`	`false`	The ASCII value of `'R'` is 82, and the ASCII value of `'T'` is 84. Because `82 > 84` is `false`, it follows that `'R' > 'T'` is `false`.
`'+' < '*'`	`false`	The ASCII value of `'+'` is 43, and the ASCII value of `'*'` is 42. Because `43 < 42` is `false`, it follows that `'+' < '*'` is `false`.
`'6' <= '>'`	`true`	The ASCII value of `'6'` is 54, and the ASCII value of `'>'` is 62. Because `54 <= 62` is `true`, it follows that `'6' <= '>'` is `true`.

سبق وتحدثنا ان نوع البيانات string هو عبارة عن مجموعة من الرموز .. يتم مقارنة قيمة الاسكي كود للرمز الاول بقيمة الاسكي كود للرمز الاول بالعنصر االثاني .. بعد ذلك ننتقل الى المز الذي يلي وهكذا حتى انتهاء رموز المتغير مثال للتوضيح ..(سلايد ١١ -١٤-شابتر ٤)

```
string str1 = "Hello"; ||string str2 = "Hi";||string str3 = "Air";||string str4 = "Bill";
string str4 = "Big";
```

TABLE 4-3 Evaluating Logical Expressions with `string` Variables

Expression	Value	Explanation
`str1 < str2`	`true`	`str1 = "Hello"` and `str2 = "Hi"`. The first characters of `str1` and `str2` are the same, but the second character `'e'` of `str1` is less than the second character `'i'` of `str2`. Therefore, `str1 < str2` is `true`.
`str1 > "Hen"`	`false`	`str1 = "Hello"`. The first two characters of `str1` and `"Hen"` are the same, but the third character `'l'` of `str1` is less than the third character `'n'` of `"Hen"`. Therefore, `str1 > "Hen"` is `false`.
`str3 < "An"`	`true`	`str3 = "Air"`. The first characters of `str3` and `"An"` are the same, but the second character `'i'` of `"Air"` is less than the second character `'n'` of `"An"`. Therefore, `str3 < "An"` is `true`.

TABLE 4-4 Evaluating Logical Expressions with `string` Variables

Expression	Value	Explanation
`str4 >= "Billy"`	`false`	`str4 = "Bill"`. It has four characters and `"Billy"` has five characters. Therefore, `str4` is the shorter string. All four characters of `str4` are the same as the corresponding first four characters of `"Billy"`, and `"Billy"` is the larger string. Therefore, `str4 >= "Billy"` is `false`.
`str5 <= "Bigger"`	`true`	`str5 = "Big"`. It has three characters and `"Bigger"` has six characters. Therefore, `str5` is the shorter string. All three characters of `str5` are the same as the corresponding first three characters of `"Bigger"`, and `"Bigger"` is the larger string. Therefore, `str5 <= "Bigger"` is `true`.

Logical (Boolean) Operators and Logical Expressions

المعاملات والتعابير المنطقية :-

يمكن للمعاملات المنطقية ان تحتوي على تعابير منطقية .."طبعاً الاشي المنطقي حالتين (true or false) .

المعاملات المنطقية (سلايد ١٥-شابتر ٤)

TABLE 4-5 Logical (Boolean) Operators in C++

Operator	Description
`!` ← unary	`not`
`&&` ← binary	`and`
`||` ← binary	`or`

المعامل الاول (!) "not":-

من اسمو not يعني "مش" الاشي الي مش غلط بكون صح والاشي الي مش صح بكون غلط .. وهو احادي يحتاج لعنصر واحد فقط

unary:- يعني احادي يحتاج عنصر واحد وليس عنصرين مثال ..

!true =false // !false=true //!(10<5)➔true

المعامل الثاني(&&)"and":-

يعني (و) ويستخدم للربط بين عنصرين يجب ان يكون ناتج العنصرين صحيح ليكون ناتج التعبير true في حال كان الطرف الاول ناتجو false لا يتم النظر الا الطرف الاخر وكانه مش موجود"بشكل منطقي لن تؤثر نتيجة العنصر الثاني بما ان الاول خطأ ف سيكون ناتج التعبير كامل flase. مثال :-

(100<200)&&(10<20)➔true// (100>200)&&(5<10)➔false

المعامل الثالث (||) "or" :-

يعني (أو) ويستخدم للربط بين عنصرين يجب ان يكون واحد منهم ناتجه true ليكون ناتج التعبير true وفي حال كان العنصر الاول ناتجه true يتم اهمال الطرف الثاني وكانه مش موجود ولا حتى بنشوفه "بشكل منطقي ايضاً لن تؤثر نتيجة العنصر الثاني على النتيجة كاملة لانه يكفي ان يكون احدهما true ليكون الناتج الكلي true مثال :-.

(11>5)||(100>200)➔true//(100>200)||(100>50)➔true//

EXAMPLE 4-3

Expression	Value	Explanation
`(14 >= 5) && ('A' < 'B')`	`true`	Because `(14 >= 5)` is `true`, `('A' < 'B')` is `true`, and `true && true` is `true`, the expression evaluates to `true`.
`(24 >= 35) && ('A' < 'B')`	`false`	Because `(24 >= 35)` is `false`, `('A' < 'B')` is `true`, and `false && true` is `false`, the expression evaluates to `false`.

Order of Precedence

اولويات العمليات .الجدول حفظ

TABLE 4-9 Precedence of Operators

Operators	Precedence
!, +, – (unary operators)	first
*, /, %	second
+, –	third
<, <=, >=, >	fourth
==, !=	fifth
&&	sixth
||	seventh
= (assignment operator)	last

example:-

EXAMPLE 4-5

Suppose you have the following declarations:

```
bool found = true;
bool flag = false;
int num = 1;
double x = 5.2;
double y = 3.4;
int a = 5, b = 8;
int n = 20;
char ch = 'B';
```

Expression	Value	Explanation
`!found`	`false`	Because `found` is `true`, `!found` is `false`.
`x > 4.0`	`true`	Because `x` is `5.2` and `5.2 > 4.0` is `true`, the expression `x > 4.0` evaluates to `true`.
`!num`	`false`	Because `num` is `1`, which is nonzero, `num` is `true` and so `!num` is `false`.
`!found && (x >= 0)`	`false`	In this expression, `!found` is `false`. Also, because `x` is `5.2` and `5.2 >= 0` is `true`, `x >= 0` is `true`. Therefore, the value of the expression `!found && (x >= 0)` is `false && true`, which evaluates to `false`.
`!(found && (x >= 0))`	`false`	In this expression, `found && (x >= 0)` is `true && true`, which evaluates to `true`. Therefore, the value of the expression `!(found && (x >= 0))` is `!true`, which evaluates to `false`.
`x + y <= 20.5`	`true`	Because `x + y = 5.2 + 3.4 = 8.6` and `8.6 <= 20.5`, it follows that `x + y <= 20.5` evaluates to `true`.

**اي شي قيمته غير الصفر =true

**الصفر =false

Selection: if and if...else

جملة الشرط if تعني إذا ..تستخدم في حالة وجود جملة يتوقف تنفيذها على شرط ..اذا كان الشرط ناتجه true . تتنفذ الجملة غير ذلك بنتركها وكانها مش هون ..

النوع الاول من جملة الشرط (if)

One-Way Selection

The syntax of one-way selection is:

```
if (expression)
    statement
```

يتم تنفيذ الجملة في حال كان ناتج التعبير الموجود بين القوسين true ...

يتم تجاهل الجملة وكانها غير موجودة في حال كان ناتج التعبير false

تستخدم في حال كان تنفيذ الشرط له حالة واحدة ..من اسمها one way مثال :-

```
if(10>2)
cout<<"true"<<endl;
```

سيتم طباعة true على شاشة المخرجات

```
if(100>200)
cout<<"some_thing"<<endl;
```

ستظهر شاشة فارغة لانو ناتج التعبير false

حالات يجب الانتباه لها :-

```
if 100<50
cout<<"some thing"<<endl;
```

ستظهر رسالة الخطأsyntax error بسبب عدم وضع التعبير بين قوسين

```
if(100>50);
cout<<"some thing"<<endl;
```

وضع اشارة الفاصلة المنقوطة بعد الشرط لا يسبب ظهور رسالة الخطأ لكن في هذه الحالة تنتهي جملة if عند الفاصلة المنقوطة وسيتم طباعة الجملة التي تليها على كل الحالات سواء كان الشرط true or false

---

example [important]

EXAMPLE 4-10

The following C++ program finds the absolute value of an integer:

```
//Program: Absolute value of an integer

#include <iostream>

using namespace std;

int main()
{
    int number, temp;

    cout << "Line 1: Enter an integer: ";          //Line 1
    cin >> number;                                 //Line 2
    cout << endl;                                  //Line 3

    temp = number;                                 //Line 4

    if (number < 0)                                //Line 5
        number = -number;                          //Line 6

    cout << "Line 7: The absolute value of "
         << temp << " is " << number << endl;      //Line 7

    return 0;
}
```

Sample Run: In this sample run, the user input is shaded.

```
Line 1: Enter an integer: -6734
Line 7: The absolute value of -6734 is 6734
```

EXAMPLE 4-9

```
if (score >= 60)
    grade = 'P';
```

In this code, if the expression (`score >= 60`) evaluates to `true`, the assignment statement, `grade = 'P';`, executes. If the expression evaluates to `false`, the statements (if any) following the `if` structure execute. For example, if the value of `score` is 65, the value assigned to the variable `grade` is `'P'`.

Two-Way Selection

النوع الثاني من جملة الشرط if في حال وجود جملتين احدهما يتنفذ في حال كان ناتج التعبير true والاخر يتنفذ في حال كان الناتج false

```
if (expression)
    statement1
else
    statement2
```

If expression is true, statement1 is executed; otherwise, statement2 is executed

---

if (100>30)
cout<<"true"<<endl;
else
cout<<"false"<<endl;
مخرجات هذا البرنامج true بسبب تحقيق الشرط

---

if(100<30)
cout<<"true"<<endl;
else
cout<<"false"<<endl;
مخرجات هذا البرنامج false بسبب عدم تحقيق الشرط الاول فيتم تنفيذ جملة else

---

ملاحظات مهمة جداً جداً جداً ..

في حال كان الشرط سينفذ اكثر من جملة يجب وضع الجمل بين قوسين راقصين { } وفي حال عدم وضعهم ستكون الجملة الاولى فقط هي التابعة والمتصلة في جملة الشرط if ولا تظهر رسالة الخطأ syntax errorفي حالة one way ..

في حالة ال two way ووجود اكثر من جملة يجب ايضاً وضعهم في قوسين راقصين وفي حال عدم وضعهم ستظهر رسالة الخطأ وذلك لسبب بسيط .. تصبح else غير متصلة ب if وهذا غير شرعي حيث ان else لا يمكن لها ان تكون منفردة بالبرنامج ..

لكتابة اكثر من جملة تحت الشرط او تحت else يجب اتباع هذا القالب :-

```
{
    statement1
    statement2
      .
      .
      .
    statementn
}
```

example:-

```
if (age > 18)
{
cout << "Eligible to vote." << endl;
cout << "No longer a minor." << endl;
}
else
{       cout << "Not eligible to vote." << endl;
cout << "Still a minor." << endl;
}
```

في حال كان المتغير age قيمتو اكبر من ١٨ سيتم تنفيذ الجملتين :-

Eligible to vote
No longer a minor
وفي حال لم يكن
Not eligible to vote.
Still a minor

Multiple Selections: Nested if

في حال وجود اكثر من خيارين نسخدم هذا النوع . بصورة عامة هذا الشكل

EXAMPLE 4-18

Suppose that `balance` and `interestRate` are variables of type `double`. The following statements determine the `interestRate` depending on the value of the `balance`:

```
if (balance > 50000.00)                  //Line 1
    interestRate = 0.07;                 //Line 2
else                                     //Line 3
    if (balance >= 25000.00)             //Line 4
        interestRate = 0.05;             //Line 5
    else                                 //Line 6
        if (balance >= 1000.00)          //Line 7
            interestRate = 0.03;         //Line 8
        else                             //Line 9
            interestRate = 0.00;         //Line 10
```

مبدأ عمل جملة الشرط في حال وجود اكثر من خيارين .
يتم فحص الشرط الاول في حال تحقيقه يتم اهمال كل ما تحته !! يعني مثل else ..وفي حال كان الشرط الاول false ننتقل الى الذي يلي وهكذا حتى يتم تحقيق شرط يتم تنفيذ جملو والخروج من جملة الشرط . وفي حال عدم تحقيق اي شرط نذهب الى else في حال وجودها وننفذ الجمل الخاصة فيها ..

واحد من اهم الامثلة ..

EXAMPLE 4-19

Assume that score is a variable of type int. Based on the value of score, the following code outputs the grade:

```
if (score >= 90)
    cout << "The grade is A." << endl;
else if (score >= 80)
    cout << "The grade is B." << endl;
else if (score >= 70)
    cout << "The grade is C." << endl;
else if (score >= 60)
    cout << "The grade is D." << endl;
else
    cout << "The grade is F." << endl;
```

اذا كان العلامة اكبر او تساوي ٩٠ سيتم تنفيذ الجملة الاولى والخروج من الجملة the grade is A

اذا كانت العلامة لا تحقق الشرط الاول ننتقل الى الثاني ونفحص الشرط اذا تحقق بنفذ جملة the grade is B

في حال لم يتحقق بننزل على الي تحتو وبنفحصو وهكذا .. في حال لم يتحقق اي من الشروط نخرج الى else وبنطبع the grade is F

a.
```
if (month == 1)                          //Line 1
    cout << "January" << endl;           //Line 2
else if (month == 2)                     //Line 3
    cout << "February" << endl;          //Line 4
else if (month == 3)                     //Line 5
    cout << "March" << endl;             //Line 6
else if (month == 4)                     //Line 7
    cout << "April" << endl;             //Line 8
else if (month == 5)                     //Line 9
    cout << "May" << endl;               //Line 10
else if (month == 6)                     //Line 11
    cout << "June" << endl;              //Line 12
```

** مهم جداً ايضاً

يرجى الانتباه هناك فرق بين سلسلة جمل الشرط وجمل الشرط النوع الثالث في حال كان اكثر من خيارين ..

b.
```
if (month == 1)
    cout << "January" << endl;
if (month == 2)
    cout << "February" << endl;
if (month == 3)
    cout << "March" << endl;
if (month == 4)
    cout << "April" << endl;
if (month == 5)
    cout << "May" << endl;
if (month == 6)
    cout << "June" << endl;
```

هنا سلسلة من جمل الشرط ..في حال كان الاول صحيح يتم النظر الى الثاني والثالث لانها تعتبر كل جملة مستقلة عن الاخرى ،،يعني لما تشوف شرط تحقق كمل بهاي الحالة ^^

Conditional Operator (?:)

معامل الشرط يستخدم (اختياري) في حال كانت جملة if بسيطة .. اي لها جملة واحدة مثال

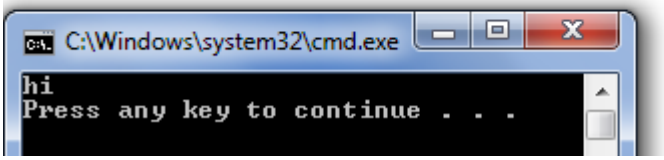
```
if (x>4)
cout<<"hey";
else
cout<<"bay";
```

يتم تحويلها الى معامل الشرط كما يلي

```
x>4?cout<<"hey" : cout<<bay;
```

مهمة بالعادة بيجي عليها سؤال بالامتحان الاول ^^

```
#include<iostream>
using namespace std;
void main()
{10>5?cout<<"hi"<<endl:cout<<"by"<<endl;
}
```



بعد علامة الاستفهام يتم وضع جملة الشرط في حال كان صح ..وبعد النقطتين الراسيتين يتم وضع جملة else

switch Structures

تستخدم للتحكم ايضاً وهي النوع الثاني من جمل التحكم بعد جملة الشرط if

```
switch (expression)
{
case value1:
    statements1
    break;
case value2:
    statements2
    break;
    .
    .
    .
case valuen:
    statementsn
    break;
default:
    statements
}
```

expression:-هذا التعبير الذي يوضع بين القوسين في جملة switch .ويجب ان يكون integral مثل int ,char, ويمكن ان يكون رقم صحيح ثابت مثل 5,10,11

case:- باللغة العربي تعني حالات ..وهي الحالات التي تتنفذ في حال كان ال exp مساوي لل case

break:- تستخدم للخروج من الجملة بعد تنفيذ الحالة .وفي حال عدم وجودها يتم التنفيذ كل الحالات التي تلي الحالة الصحيحة .

default :- في حال عد توافق اي حالة مع ال exp يتم تنفيذ ال default

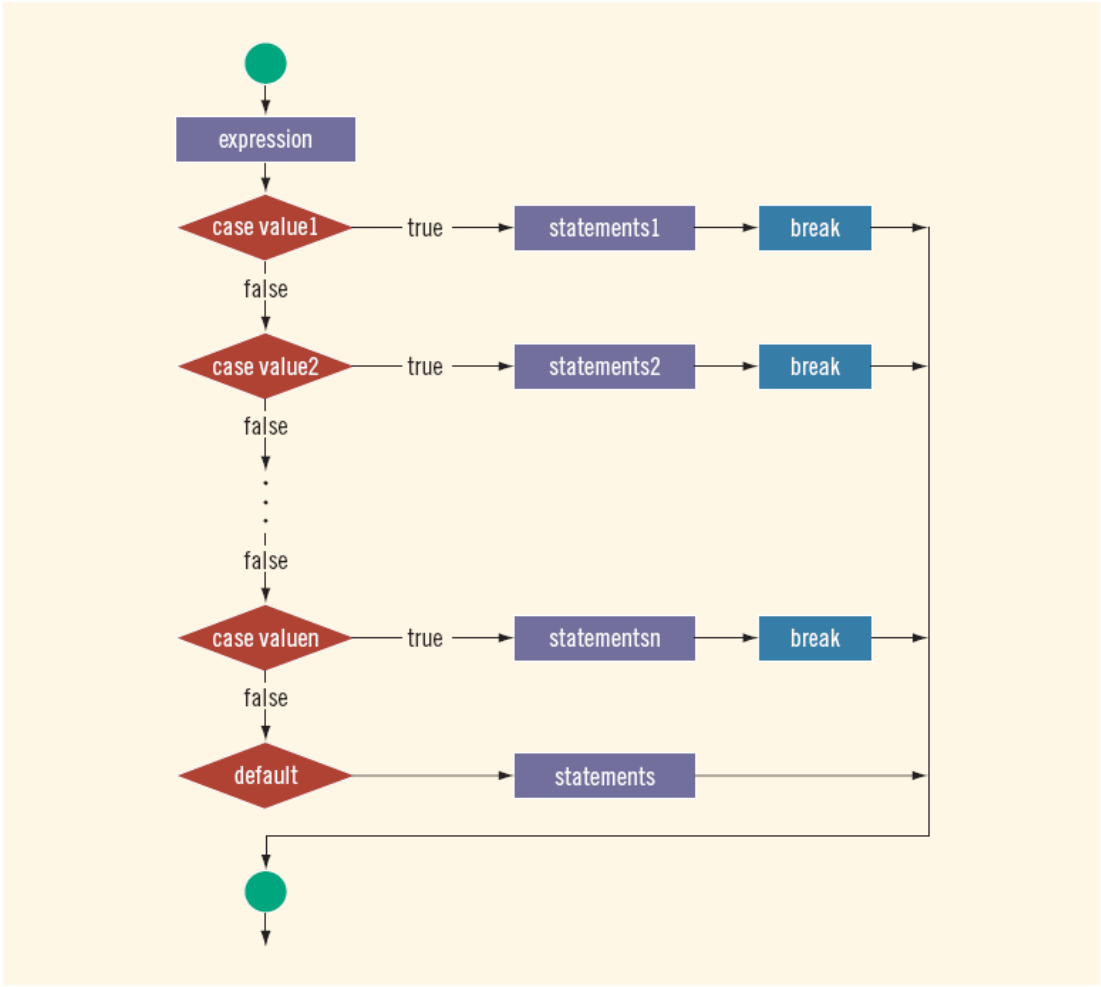


FIGURE 4-4 `switch` statement

الرسمة مهمة جداً وتوضح عملية مسار ال switch ...

EXAMPLE 4-24

Consider the following statements, where grade is a variable of type `char`:

```
switch (grade)
{
case 'A':
    cout << "The grade is 4.0.";
    break;
case 'B':
    cout << "The grade is 3.0.";
    break;
case 'C':
    cout << "The grade is 2.0.";
    break;
case 'D':
    cout << "The grade is 1.0.";
    break;
case 'F':
    cout << "The grade is 0.0.";
    break;
default:
    cout << "The grade is invalid.";
}
```

In this example, the expression in the `switch` statement is a variable identifier. The variable `grade` is of type `char`, which is an integral type. The possible values of `grade` are `'A'`, `'B'`, `'C'`, `'D'`, and `'F'`. Each `case` label specifies a different action to take, depending on the value of `grade`. If the value of `grade` is `'A'`, the output is:

```
The grade is 4.0.
```

what the output

#include<iostream>
using namespace std;
void main()
{int x=5;
if(x>=5)
cout<<"first "<<x+1<<endl;
if (x==6)
cout<<"second"<<x-1<<endl;
}
	first 6

#include<iostream>
using namespace std;
void main()
{int x=3,y=5;
bool flag=false;
//flag=!flag;
cout<<(flag||x-2&&y==8)<<endl;
}
	0

#include<iostream>
using namespace std;
void main()
{int x=3,y=7;
if (x<=3)
cout<<x+y<<endl;
cout<<x-y<<endl;
}
	10
	-4

#include<iostream> using namespace std; void main() {int a=5,b,c=6; b=2 + ++c * a++; cout<<a<<"\t"<<b<<"\t"<<c<<endl;}	6 37 7
cout<<"\"my\\home is\beside\'her home\\b"<<endl;	"my\home ieside ' her home\b

if (static_cast<int>(6.9)-3 >=3.9)
cout<<"cs101"<<endl;
cout<<"cs102"<<endl;
if (static_cast<bool>('A')=='A')
cout<<"cs112"<<endl;
cout<<"cs113"<<endl;
	cs102
	cs113

ما اصبت به فمن الله وما اخطأت به فمني ومن الشيطان

م.مُحَمَّد المشرقي