Умовні оператори

- if
- switch
- ?:

Логічні операції

Операції відношення порівнюють значення **Л** зі значенням **П**:

< — менше;

<= — менше або дорівнює (не перевищує);

== — дорівнює;

> — більше;

>= — більше або дорівнює (не менше);

!= — не дорівнює.

У мові С++ «істина» — це ненульова величина, «неправда» — це нуль **(0)**. Усі не нульові значення це істина.

Операції відношення повертають ціле значення 1, якщо умова вірна, або 0, якщо умова помилкова.

Логічні операції оперують з цілими розмірами або з розмірами, які можна перетворити на цілі. Обчислення зупиняється, які тільки визначиться, чи є вираз правдивим («істина») або помилковим («неправда»). При цьому, як і для операцій відношення, значенням «істина» відповідає 1, а значенням «неправда» — 0.

&& — логічне **«AND»** (кон'юнкція);

|| — логічне **«OR»** (диз'юнкція);

! — логічне **«NOT»** (заперечення).

Результат операції «&&» є «істина» **(1)**, якщо обидва її операнди правдиві (не рівні 0). Результат операції «||» — «істина»**(1)**, якщо хоча б один з її операндів є «істина». Логічне заперечення «!=» перетворює свій операнд на «істину» **(1)**, якщо він дорівнює **0**, і на «неправду» **(0)**, якщо він не дорівнює **0**.

Операції обробки окремих бітів застосовують для обробки даних як послідовностей бітів (розрядів), кожний з яких набуває значення **0** або **1**.

& — операція бітового множення (кон'юнкція);

| — операція бітового додавання (диз'юнкція);

^ — додавання за модулем 2;

~ — інвертування;

>> — зсув праворуч;

<< — зсув ліворуч.

Оператор if

Загальний синтаксис умовного оператора if має наступний синтаксис

if (вираз) оператор 1
[else оператор 2]

Якщо значення виразу “істина” (не нуль), то виконується **оператор1**, якщо ж воно хибне, то виконується **оператор2.** Гілка else не обов’язкова і може бути відсутня.

Рекомендований синтаксис

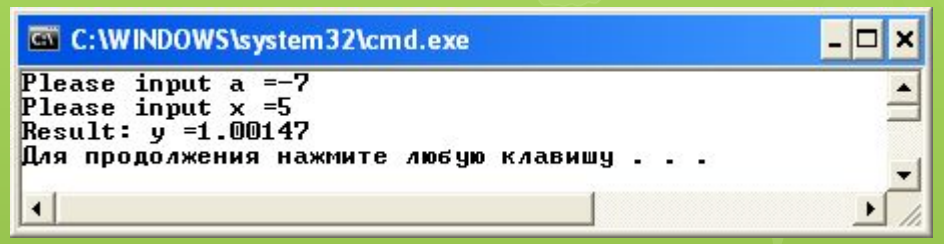
```
if (вираз)
{
    Оператори
}
else
{
    Оператори
}
```

Приклад 1

Обчислити значення виразу $y = \begin{cases} e^{x^a}, & \text{де } x > 0 \text{ і } a > 0 \\ x^{e^a} \end{cases}$

для введених з клавіатури значень для a і x.

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
using namespace std;
int main()
{
double a, x;
cout<<"Please input a =";
cin >> a;
cout<<"Please input x =";
cin >> x;
double y;
if (x > 0 && a>0)
{
     y = exp(pow(x,a));
}
else
{
     y = pow(x,exp(a));
}
cout<< "Result: y ="<<y<<endl;
system("pause");
return 0;
}
```



Складене розгалуження if

Складне розгалуження безпосередньо мовою С не підтримується.

Воно реалізовується як комбінація простих розгалужень.

Рекомендований синтаксис складного розгалуження

```
if (вираз)
{
    Оператори
}
else if (вираз)
{
    Оператори
}
else if (вираз)
{
    Оператори
}
……
else
{
    Оператори
}
```

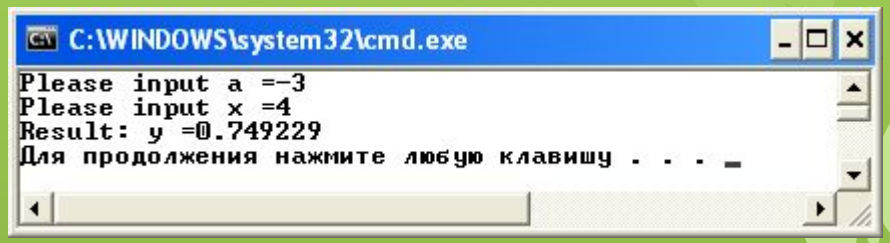
Приклад 2

Обчислити значення виразу $$y = \begin{cases} \sin|a^x|, & a > 0 \text{ і } x > 0, \\ \cos|e^x|, & a > 0, x < 0; \\ \sin(x) * \cos(a) \end{cases}$$

для введених з клавіатури значень для a і x.

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
using namespace std;
int main()
{
double a, x;
cout<<"Please input a =";
cin >> a;
cout<<"Please input x =";
cin >> x;
double y;
if (a > 0 && x>0)
{
      y = sin(fabs(pow(a,x)));
}
else if (a > 0 && x<0)
{
      y = cos(fabs(exp(x)));
}
else
{
      y = sin(x)*cos(a);
}
cout<< "Result: y ="<<y<<endl;
system("pause");
return 0;
}
```

Приклад роботи програми



Оператор вибору **switch**

загальний синтаксис

Загальний синтаксис умовного оператора вибору **switch** має наступний вигляд

```
switch (вираз)
{
case константне_значення_1: оператори
case константне_значення_2: оператори
……
константне_значення_N: оператори
default: оператори
}
```

Оператор switch порівнює значення виразу з набором константних значень. Якщо відповідність знайдена то будуть виконуватися усі оператори з відповідного case до кінця оператора switch або поки не «зустріне» оператор break. Якщо відповідність не знайдена, то буде виконуватися гілка default, якщо вона присутня, оскільки вона не обов'язкова.

Оператор вибору **switch**

рекомендований синтаксис

Якщо необхідно реалізувати для кожного case виконання тільки одного набору операторів, у цьому випадку необхідно в кінці кожного case поставити оператор break. Рекомендований синтаксис виглядає наступним чином (кожна гілка оформлена додатковим блоком):

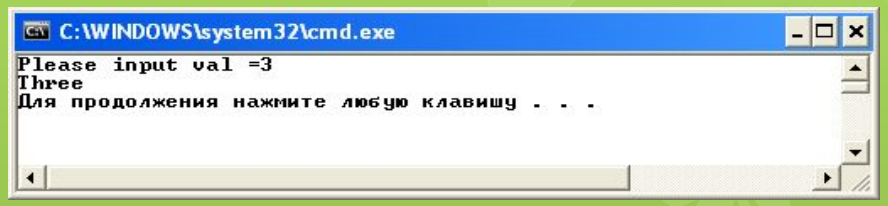
```
switch (вираз)
{
case константне_значення_1:
                                {
                                    деякий код
                                    break;
                                }
case константне_значення_2:
                                {
                                    деякий код
                                    break;
                                }
……
константне_значення_N:
                                {
                                    деякий код
                                    break;
                                }
default:
                                {
                                    деякий код
                                }
}
```

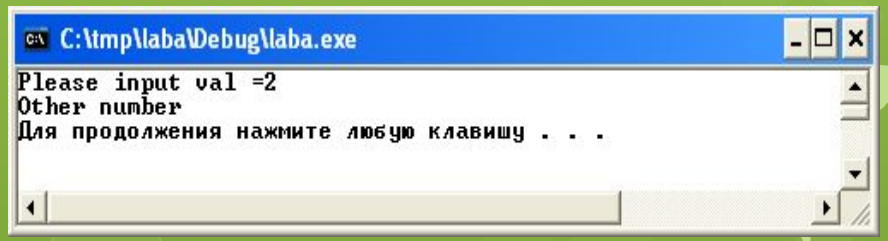
Приклад 3

Для введених чисел 1, 3, 5 надрукувати їх словесний еквівалент (One, Three, Five) у інших випадках надрукувати “Other number”.

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
using namespace std;
int main()
{
int val;
cout<<"Please input val =";
cin>>val;
switch (val)
{
     case 1: cout<< "One"      <<endl; break;
     case 3: cout<< "Three"    <<endl; break;
     case 5: cout<< "Five"     <<endl; break;
     default : cout<< "Other number"<<endl;
}
system("pause");
return 0;
}
```

Приклади роботи програми



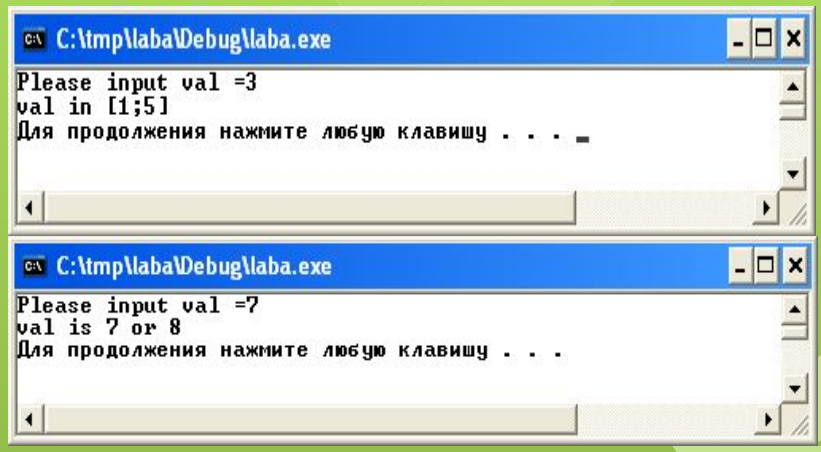


Приклад 4

Для введених цілих чисел від 1 до 5 надрукувати повідомлення “val in [1;5]”, для чисел від 7,8 надрукувати повідомлення “val is 7 or 8”. У інших випадках надрукувати “Other number”

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
using namespace std;
int main()
{
	int val;
	cout<<"Please input val =";
	cin>>val;
	switch (val)
	{
		case 1:
		case 2:
		case 3:
		case 4:
		case 5:
			cout<< "val in [1;5]"<<endl;
			break;
		case 7:
		case 8:
			cout<< "val is 7 or 8"<<endl;
			break;
		default : cout<< "Other number"<<endl;
	}
	system("pause");
	return 0;
}
```

Приклади роботи програми

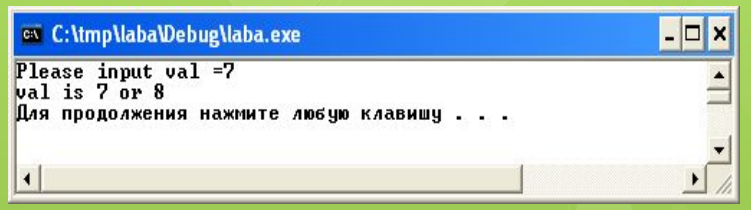


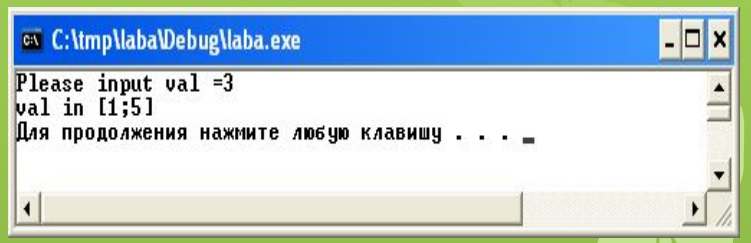
Приклад 5

Організувати виведення словесного еквіваленту від числа від Для введених цілих чисел від 1 до 5 надрукувати повідомлення “val in [1;5]”, для чисел від 7,8 надрукувати повідомлення “val is 7 or 8”. У інших випадках надрукувати “Other number”.

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
using namespace std;
int main()
{
	int val;
	cout<<"Please input val =";
	cin>>val;
	switch (val)
	{
		case 1:
		case 2:
		case 3:
		case 4:
		case 5:
			cout<< "val in [1;5]"<<endl;
			break;
		case 7:
		case 8:
			cout<< "val is 7 or 8"<<endl;
			break;
		default : cout<< "Other number"<<endl;
	}
	system("pause");
	return 0;
}
```

Приклади роботи програми





Тернарна операція ?:

Загальний синтаксис тернарної операції має наступний синтаксис

```
(вираз 1)? вираз2: вираз3
```

Вираз 1 задає умову, якщо вираз1 істинний то буде підставлений вираз 2, у іншому випадку буде підставлений вираз 3.

Наприклад, якщо необхідно знайти мінімальне значення серед змінних x та y, то це можна реалізувати наступним чином:

```
min = (x<y)? x : y;
```

в результаті min отримає значення x якщо воно меньше y, або y якщо умова у дужка не виконається.

Аналогічне вирішення задачі находження мінімума через умовний оператор if має наступний вигляд:

```
if (x<y)
     min = x;
else
     min = y;
```

Приклад 6

Вирішити задачу з прикладу 1 через тернарну операцію

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
using namespace std;
int main()
{
double a, x;
cout<<"Please input a =";
cin >> a;
cout<<"Please input x =";
cin >> x;
double y  =(x > 0 && a>0)? exp(pow(x,a)) :  pow(x,exp(a));
cout<< "Result: y ="<<y<<endl;
system("pause");
return 0;
}
```

$$y=\begin{cases} e^{x^a}, & де\ x>0\ і\ a>0 \\ x^{e^a} \end{cases}$$

Приклад роботи програми