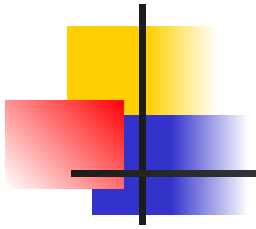
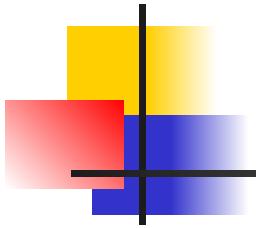


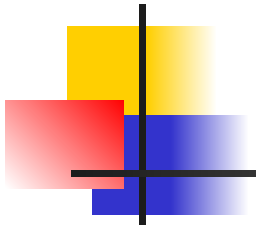
Глава 5

Условные операторы



Задачи:

- Объяснить конструкцию выбора:
  - Оператор If
  - Конструкция If – else
- Объяснить оператор перехода



Условные конструкции

- Условные конструкции позволяют нам менять ход программы
- Условные конструкции могут принимать два значения “истинно” или “ложно”

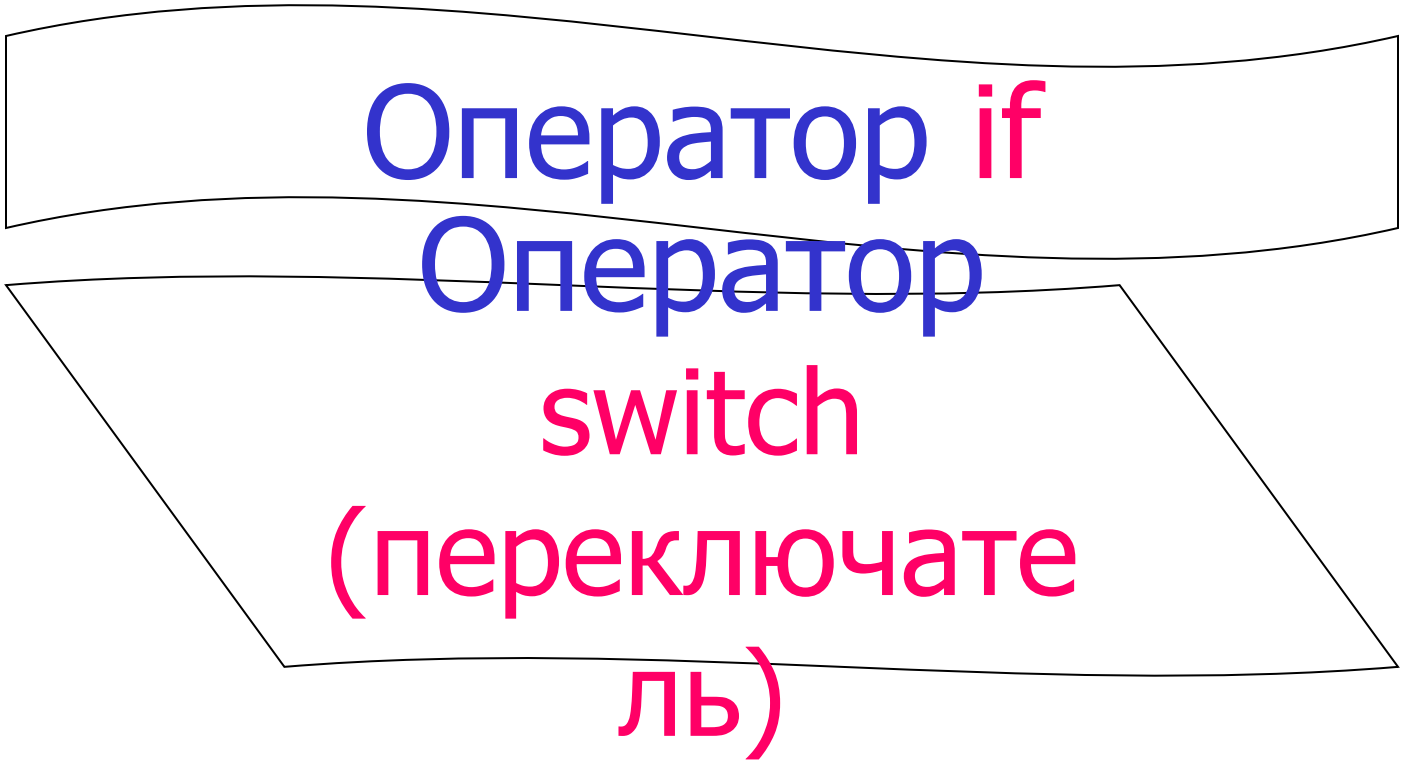
Пример :

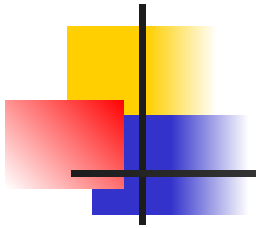
Чтобы определить четное число или нет, сделаем следующее :

1. **Введем число**
2. **Найдем остаток с помощью деления числа на 2**
3. **Если остаток равен нулю, то число четное**
4. **Если остаток не равен нулю, то число не четное**

Конструкции выбора

Язык С поддерживает два типа конструкций выбора





Оператор if

Синтаксис:

If (выражение)

оператор;

Если оператор if принимает значение true (истина), то исполняется блок ниже расположенных команд или утверждений или блока утверждений

Оператор if

Программа, выводящая значения основанные на условии

Пример

```
#include <stdio.h>
int main()
{
    int x, y;
    char a = 'y';
    x = y = 0;
    if (a == 'y')
    {
        x += 5;
        printf("The numbers are %d and \t%d", x,
y);
    }
}
```

Конструкция if – else-1

Синтаксис:

If (выражение)

оператор;

Else

оператор;

Конструкция if – else

- Если оператор if принимает значение true(истина), то исполняется блок ниже расположенных команд или утверждений или блока утверждений

- Если оператор if принимает значение false(ложь), то исполняется блок следующий по ветке else

- Ветка else может присутствовать, а может и нет, это зависит от программы. Она используется, если утверждение или блок утверждений принимает значение false(ложь)

Конструкция if – else

Программа отображающая, является ли число четным или нет

Пример

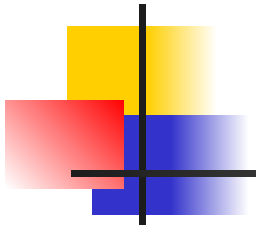
```
#include <stdio.h>
int main()
{
    int num , res ;

    printf("Enter a number :");
    scanf("%d",&num);
    res = num % 2;
    if (res == 0)
        printf("Then number is Even");
    else
        printf("The number is Odd");
}
```

Конструкция ветвления if-1

- **if** это **if** оператор, который может быть помещен между другим **if** или **else**

- В языке C++ оператор **else** всегда относится к ближайшему **if**

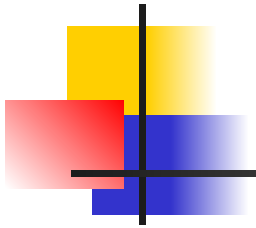
Переключатель (switch)



- Оператор **конструкции перехода** дает возможность выбора пути в зависимости от значения оцениваемого выражения statement и проверки его на значение из списка целых чисел или знаков

- Когда совпадение найдено, исполняются выражения ассоциируемые с этой константой

Переключатель



Синтаксис:

```
Switch (выражение)
{
    Case КОНСТАНТА1:
        оператор
        break;
    Case КОНСТАНТА2:
        оператор
        break;
…..
Default:
    оператор}
```

Переключатель

**Программа проверяет, является ли введенный символ гласной букво
или z или согласной**

Пример

```
#include <stdio.h>
    int main ()
    {
        char ch;
        clrscr ();

        printf ("\nEnter a lower cased alphabet (a - z) :
");
        scanf("%c", &ch);
```

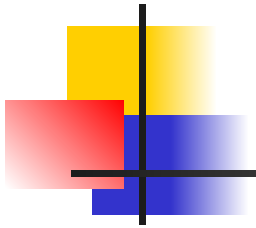
contd…….

Переключатель

```
if (ch < 'a' || ch > 'z')
    printf("\nCharacter not a lower cased alphabet");
else
    switch (ch)
    {
        case 'a' :
        case 'e' :
        case 'i' :
        case 'o' :
        case 'u' :
            printf("\nCharacter is a vowel");
            break;
        case 'z' :
            printf ("\nLast Alphabet (z) was entered");
            break;
        default :
            printf("\nCharacter is a consonant");
            break;
            }
    }
```

Пример

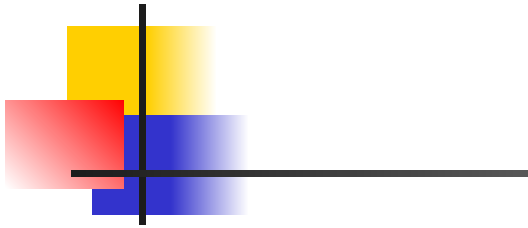
Условный тернарный оператор



Условный (тернарный) оператор (обозначается как **?:**) является единственным тернарным оператором в языке C++, который работает с 3-мя операндами. Из-за этого его часто называют просто «тернарный оператор».

Оператор	Пример	Символ	Операция
Условный	?:	c ? x : y	Если c — ненулевое значение (true), то вычисляется x, в противном случае — y

Условный тернарный оператор



Оператор **?:** предоставляет сокращенный способ (альтернативу) ветвления if/else.

Конструкцию if/else:

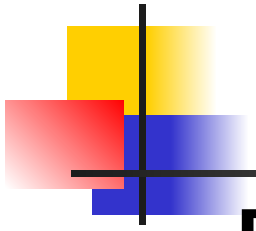
```
if (условие)
        выражение;
else
        другое_выражение;
```

Можно записать как:

```
(условие) ? выражение : другое_выражение;
```

Обратите внимание, операнды условного оператора должны быть **выражениями** (а не другими конструкциями).

Условный тернарный оператор



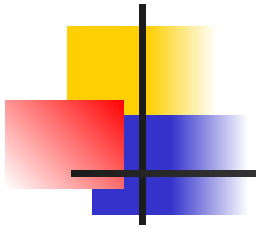
Пример 1:
ветвление if/else, которое выглядит следующим образом:

```
if (условие)
        x = значение1;
else
        x = значение2;
```

Можно записать как:

```
x = (условие) ? значение1 : значение2;
```

Условный тернарный оператор



Оператор **?:** имеет очень низкий приоритет, из-за этого его следует записывать в круглых скобках.

Например, для вывода x или y, мы можем сделать следующее:

```
if (x > y)
    std::cout << x;
else
    std::cout << y;
```

Или с помощью тернарного оператора:

```
std::cout << ((x > y) ? x : y);
```

Условный тернарный оператор

Если мы не заключим в скобки весь условный оператор в вышеприведенном случае. Поскольку оператор **<<** имеет более высокий приоритет, чем оператор **?:**, то следующее выражение:

```
std::cout << (x > y) ? x : y;
```

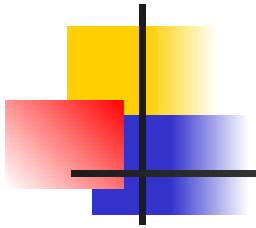
Будет обрабатываться как:

```
(std::cout << (x > y)) ? x : y;
```

Таким образом, в консольном окне мы увидим **1** (true), если x > y, в противном случае — выведется **0** (false).

Совет: **Всегда заключайте в скобки условную часть тернарного оператора, а лучше весь тернарный оператор.**

Оператор **goto**



Оператор **goto** — это оператор управления потоком выполнения программ, который заставляет центральный процессор выполнить переход из одного участка кода в другой (осуществить прыжок).

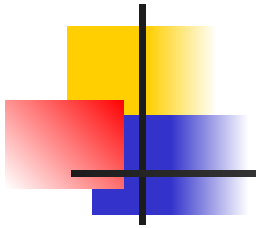
Другой участок кода идентифицируется с помощью **метки**.

Оператор **goto**

Пример:

```
#include <iostream>
#include <cmath> // для функции sqrt()

int main()
{
    double z;
tryAgain: // это метка
    std::cout << "Enter a non-negative number: ";
    std::cin >> z;

    if (z < 0.0)
        goto tryAgain; // а это оператор goto

    std::cout << "The sqrt of " << z << " is " << sqrt(z) << std::endl;
    return 0;
}
```

Оператор **goto**

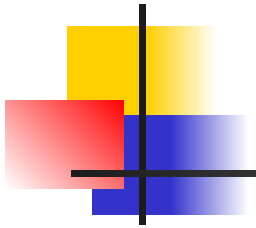


В целом, программисты избегают использования оператора **goto** в языке C++ (и в большинстве других высокоуровневых языков программирования).

Основная проблема с ним заключается в том, что он позволяет программисту управлять выполнением кода так, что точка выполнения может прыгать по коду произвольно.

«**Качество программистов — это уменьшающаяся функция плотности использования операторов goto в программах, которые они пишут**».
Эдсгер Дейкстра.

Оператор **go to**



Оператор **goto** в языке C++ почти никогда не используется, поскольку любой код, написанный с ним, можно более эффективно переписать с использованием других объектов в языке C++, таких как **циклы**, **обработчики исключений** или **деструкторы** .

Правило: **Избегайте использования операторов goto, если на это нет веских причин.**