УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите приведенную лекцию, законспектируйте основные тезисы и основные правила оптимизации.

Ответы на вопросы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 20.02.2023.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция № 7

Тема: «Условные операторы С++»

1. ***Условные ветвления if/else***
2. ***Использование нескольких операций в ветвлениях if/else***
3. ***Неявное указание блоков***
4. ***Связывание стейтментов if***
5. ***Вложенные ветвления if/else***
6. ***Использование логических операторов в ветвлениях if/else***
7. ***Основные использования ветвлений if/else***

Условные ветвления if/else

Самыми простыми условными ветвлениями в языке С++ являются стейтменты if/else. Они выглядят следующим образом:

```
if (выражение)
    стейтмент1
```

Либо так:

```
if (выражение)
    стейтмент1
else
    стейтмент2
```

выражение называется **условием** (или ***«условным выражением»***). Если результатом выражения является true (любое ненулевое значение), то выполняться будет стейтмент1. Если же результатом выражения является false (0), то выполняться будет стейтмент2. Например:

```
#include <iostream>

int main()
{
    std::cout << "Enter a number: ";
    int a;
    std::cin >> a;

    if (a > 15)
        std::cout << a << " is greater than 15\n";
    else
        std::cout << a << " is not greater than 15\n";

    return 0;
}
```

Использование нескольких операций в ветвлениях if/else

Оператор if выполняет *только одну* операцию, если выражение является true, и также *только одну* операцию else, если выражение — false. Чтобы выполнить несколько операций подряд, используйте **блок стейтментов**:

```cpp
#include <iostream>

int main()
{
    std::cout << "Enter a number: ";
    int a;
    std::cin >> a;

    if (a > 15)
        {
        // Обе операции будут выполнены, если a > 15
        std::cout << "You entered " << a << "\n";
        std::cout << a << " is greater than 15\n";
        }
    else
        {
        // Обе операции будут выполнены, если a <= 15
        std::cout << "You entered " << a << "\n";
        std::cout << a << " is not greater than 15\n";
        }

    return 0;
}
```

Неявное указание блоков

Если программист не указал скобки для блока стейтментов if или else, то компилятор неявно сделает это за него. Таким образом, следующее:

```
if (выражение)
    стейтмент1
else
    стейтмент2
```

Будет выполняться как:

```
if (выражение)
{
    стейтмент1
}
else
{
    стейтмент2
}
```

По сути, это не имеет значения. Однако начинающие программисты иногда пытаются сделать что-то вроде следующего:

```cpp
#include <iostream>

int main()
{
    if (1)
        int a = 4;
    else
        int a = 5;

    std::cout << a;

    return 0;
}
```

Программа не скомпилируется, и в итоге мы получим ошибку, что идентификатор ***a*** не определен. А произойдет это из-за того, что программа будет выполняться следующим образом:

```cpp
#include <iostream>

int main()
{
    if (1)
    {
        int a = 4;
    } // переменная a уничтожается здесь
    else
    {
        int a = 5;
    } // переменная a уничтожается здесь

    std::cout << a; // переменная a здесь не определена

    return 0;
}
```

В этом контексте становится понятным, что переменная `a` имеет **локальную область видимости** и уничтожается в конце блока, в котором выполняется её инициализация. И, когда мы дойдем до строчки с **std::cout**, переменная `a` уже перестанет существовать.

Связывание стейтментов if

Стейтменты if/else можно использовать в связке:

```cpp
#include <iostream>

int main()
{
    std::cout << "Enter a number: ";
    int a;
    std::cin >> a;

    if (a > 15)
        std::cout << a << " is greater than 15\n";
    else if (a < 15)
        std::cout << a << " is less than 15\n";
    else
        std::cout << a << " is exactly 15\n";

    return 0;
}
```

Вложенные ветвления if/else

Одни стейтменты if могут быть вложены в другие стейтменты if:

```cpp
#include <iostream>

int main()
{
    std::cout << "Enter a number: ";
    int a;
    std::cin >> a;

    if (a > 15) // внешний оператор if
        // Это плохой способ написания вложенных стейтментов if
        if (a < 25) // внутренний оператор if
            std::cout << a << " is between 15 and 25\n";

        // К какому if относится следующий else?
        else
            std::cout << a << " is greater than or equal to 25\n";

    return 0;
}
```

Обратите внимание, в программе, приведенной выше, мы можем наблюдать **потенциальную ошибку двусмысленности оператора else**. К какому if относится оператор else: к внешнему или к внутреннему?

Дело в том, что оператор else всегда относится к последнему незакрытому оператору if в блоке, в котором находится сам else. Т.е. в программе, приведенной выше, else относится к внутреннему if.

Чтобы избежать таких вот неоднозначностей при вложенности операторов условного ветвления, рекомендуется использовать блоки стейтментов (указывать скобки). Например, вот та же программа, приведенная выше, но уже без двусмысленности:

```cpp
#include <iostream>

int main()
{
    std::cout << "Enter a number: ";
    int a;
    std::cin >> a;

    if (a > 15)
    {
        if (a < 25)
            std::cout << a << " is between 15 and 25\n";
        else // относится к внутреннему оператору if
            std::cout << a << " is greater than or equal to 25\n";
    }

    return 0;
}
```

Теперь понятно, что оператор else относится к внутреннему оператору if. Использование скобок также позволяет явно указать привязку else к внешнему стейтменту if:

```cpp
#include <iostream>

int main()
{
    std::cout << "Enter a number: ";
    int a;
    std::cin >> a;

    if (a > 15)
    {
        if (a < 25)
            std::cout << a << " is between 15 and 25\n";
    }
    else // относится к внешнему оператору if
        std::cout << a << " is less than 15\n";

    return 0;
}
```

Используя блоки стейтментов, мы уточняем, к какому if следует прикреплять определенный else. Без блоков оператор else будет прикрепляться к ближайшему незакрытому оператору if.

Использование логических операторов в ветвлениях if/else

Также вы можете проверить сразу несколько условий в ветвлениях if/else, используя **логические операторы**:

```cpp
#include <iostream>

int main()
{
    std::cout << "Enter an integer: ";
    int a;
    std::cin >> a;

    std::cout << "Enter another integer: ";
    int b;
    std::cin >> b;

    if (a > 0 && b > 0) // && - это логическое И. Проверяем, являются ли оба условия истинными
        std::cout << "Both numbers are positive\n";
    else if (a > 0 || b > 0) // || - это логическое ИЛИ. Проверяем, является ли истинным хоть одно из услов
        std::cout << "One of the numbers is positive\n";
    else
        std::cout << "Neither number is positive\n";

    return 0;
}
```

Основные использования ветвлений if/else

Ветвления if/else активно используются для проверки ошибок. Например, чтобы вычислить квадратный корень значения, параметр, который передается в функцию для вычисления, — обязательно должен быть положительным:

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

void printSqrt(double value)
{
    if (value >= 0.0)
        std::cout << "The square root of " << value << " is " << sqrt(value) << "\n";
    else
        std::cout << "Error: " << value << " is negative\n";
}
```

Также операторы if используют для **ранних возвратов** — когда функция возвращает управление обратно в caller еще до завершения выполнения самой функции. В программе, приведенной ниже, если значением параметра является отрицательное число, то функция сразу же возвращает в caller **символьную константу** или **перечислитель** в качестве кода ошибки:

```cpp
#include <iostream>

enum class ErrorCode
{
    ERROR_SUCCESS = 0,
    ERROR_NEGATIVE_NUMBER = -1
};

ErrorCode doSomething(int value)
{
    // Если параметром value является отрицательное число,
    if (value < 0)
        // то сразу же возвращаем код ошибки
        return ErrorCode::ERROR_NEGATIVE_NUMBER;

    // Что-нибудь делаем

    return ErrorCode::ERROR_SUCCESS;
}

int main()
{
    std::cout << "Enter a positive number: ";
    int a;
    std::cin >> a;

    if (doSomething(a) == ErrorCode::ERROR_NEGATIVE_NUMBER)
    {
        std::cout << "You entered a negative number!\n";
    }
    else
    {
        std::cout << "It worked!\n";
    }

    return 0;
}
```

Ветвления if/else также обычно используют для выполнения простых математических операций. Например, рассмотрим функцию min(), которая возвращает минимальное из 2-х чисел:

```
int min(int a, int b)
{
    if (a > b)
        return b;
    else
        return a;
}
```

Эта функция настолько проста, что её можно записать с помощью **условного тернарного оператора**:

```
int min(int a, int b)
{
    return (a > b) ? b : a;
}
```

Контрольные вопросы:

1. Условные ветвления if/else
2. Использование нескольких операций в ветвлениях if/else
3. Неявное указание блоков
4. Связывание стейтментов if
5. Вложенные ветвления if/else
6. Использование логических операторов в ветвлениях if/else
7. Основные использования ветвлений if/else