SCC0222 - Laboratório de Introdução à Ciência de Computação I

Estruturas Condicionais

Prof.: Leonardo Tórtoro Pereira

leonardop@usp.br

Estruturas Condicionais

Estrutura Condicionais

- Usada para definir qual “caminho” o código deve seguir
- Muda o fluxo de execução do programa de acordo com as condições impostas
- Deve-se tomar cuidado para garantir que todas as possibilidades sejam levadas em conta
- Pode ser dada pela estrutura *if-else-else if*, por *switch-case* ou *operador ternário*

If-Else-Else if

- A estrutura mínima é com o bloco de código
  - *if(condição) {código a ser executado}*
- Na qual a condição pode ser qualquer expressão que retorne um valor inteiro
- Se a condição for verdadeira (valor diferente de zero) o bloco de código abaixo do *if* será executado
- Caso seja falsa (valor zero), o bloco de código é ignorado

If-Else-Else if

- Condições usam os operadores condicionais
  - Igual a “==”
  - Não igual a “!=”
  - Maior que “>” ou Maior ou Igual a “>=”
  - Menor que “<” ou Menor ou Igual a “<=”
- Os operadores lógicos também podem ser usados para conectar expressões
  - AND “&&”, OR “||” e NOT “!”

If-Else-Else if

```c
int main()
{
    int a = 1;
    float b = 0.4;
    if(1 == 1)
        printf("1 é igual a 1, é verdade!\n");
    if(a != 1)
        printf("1 não é diferente de 1, é falso e não serei impresso :(\n");
    if (1 > 0.4)
        printf("1 é maior que 0.4, é verdade!\n");
    if(a < b)
        printf("1 não é menor que 0.4, é falso e não serei impresso :(\n");
    if(a <= b || a >= b)
        printf("Uma das duas tem que ser verdade\n");
    if(a == 1 && b == 0.4)
        printf("Os dois tem que ser verdade, por isso fui impresso\n");
    if(0.2 < 0.5)
    {
        printf("Também pode ser um bloco de código e não só uma linha\n");
        a = 0;
    }
}
```

If-Else-Else if

- ➔ Você pode executar um outro bloco de código caso a condição retorne falso com o comando *else {código}*
- ➔ Você também pode colocar mais de uma condição com o comando *else if(condição) {código}*
- ➔ Você pode colocar quantos *else if* quiser depois de um *if()*
- ➔ Ambos só podem ser chamados DEPOIS de um *if()*

If-Else-Else if

```c
int main()
{
    int a = 1, b = 2;
    float c = 1.5;
    char d= 'c';
    if(d == 'x')
        printf("d é um caractere x\n");
    else
        printf("d não é um caractere x\n");
    if(c == a)
        printf("c é igual a a\n");
    else if(c == b)
        printf("c é igual a b\n");
    else if(c > a)
        printf("c é maior que a\n");
    else if (b > a)
        printf("b é maior que a\n");
    else
        printf("Não faço ideia do que testar\n");
}
```

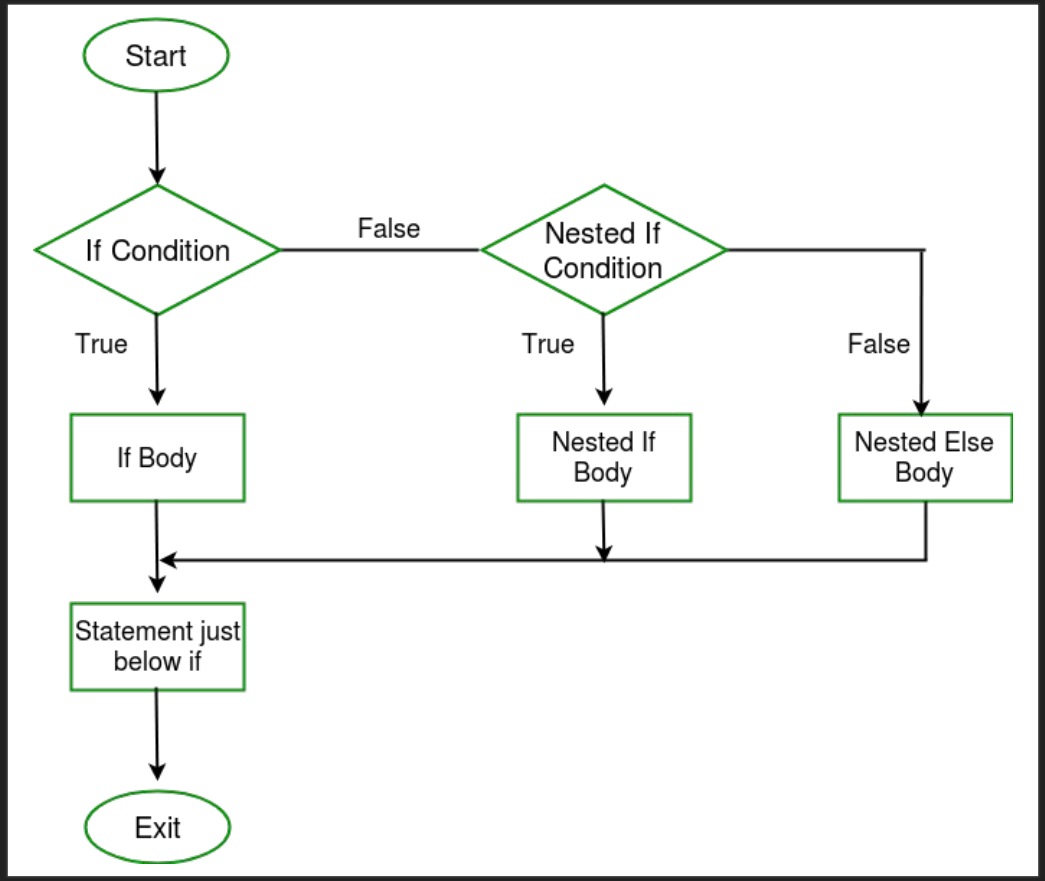
If-Else-Else if

- ➔ É possível aninhar declarações condicionais caso uma sequência de condições precise ser satisfeita
- ➔ É uma alternativa à conectar expressões com operadores lógicos

If-Else-Else if aninhados

```c
int main(){
    int teste1, teste2, teste3, teste4;
    teste1 = teste2 = teste3 = teste4 = 1;

    if(teste1 > 0)
        if(teste2 > 0)
            printf("Passou nos testes 1 e 2\n");
        else if(teste3 > 0)
            printf("Passou nos testes 1 e 3\n");
        else
            if(teste4 > 0)
                printf("Passou nos testes 1 e 4\n");
            else
                printf("Passou só no teste 1\n");
    else
        if(teste2 > 0)
            printf("Passou só no teste 2\n");
        else
            printf("Não passou em nenhum teste\n");
    return 0;
}
```



Fonte: https://www.geeksforgeeks.org/decision-making-c-c-else-nested-else/

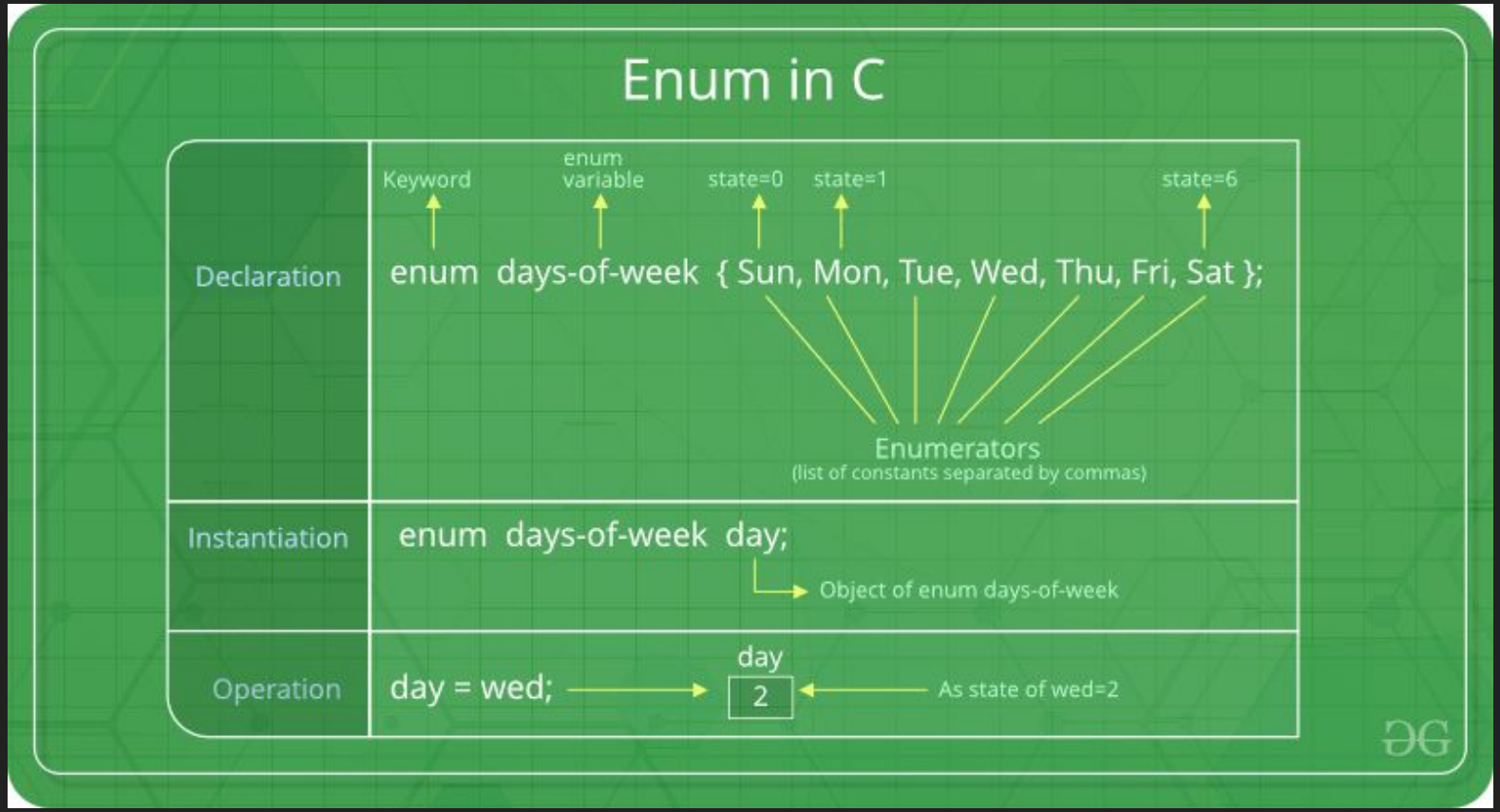
Exemplo de um Simulador de Dano

```c
enum types{Fire, Water, Wind, Earth, Light, Dark}; //0 a 5
int main() {
    int defenderHP, attackerAtk, defenderType, attackerType, damageMultiplier, hitChance, randHit;
    scanf("%d %d %d %d %d", &defenderHP, &defenderType, &attackerAtk, &attackerType, &hitChance);
    randHit = rand()%100;
    if(randHit < hitChance) {
        if(attackerType == Water && defenderType == Fire)
            damageMultiplier = 2;
        else if(attackerType == Wind && defenderType == Earth)
            damageMultiplier = 2;
        else if(attackerType == Light && defenderType == Dark)
            damageMultiplier = 2;
        else if(attackerType == Fire && defenderType == Water)
            damageMultiplier = 0.5;
        else if(attackerType == Earth && defenderType == Wind)
            damageMultiplier = 0.5;
        else if(attackerType == Dark && defenderType == Light)
            damageMultiplier = 0.5;
        else
            damageMultiplier = 1.0;
        defenderHP -= attackerAtk*damageMultiplier;
    }
    else
        printf("O ataque falhou!\n");
    return 0;
}
```

Enum? Rand()?

Enum [2]

- Um tipo de dado definido por usuário
- Muito usado para dar nomes à constantes inteiras
- Mais fácil de ler código e dar manutenção
  - Nomes no domínio da aplicação
- Pode adicionar ou remover do enum diretamente e, se feito do jeito certo, integridade do código é mantida (ou pelo menos o compilador acusa o que foi removido)



Fonte: https://www.geeksforgeeks.org/enumeration-enum-c/

Enum [2]

➔ Você pode também definir variáveis do tipo enum

```c
enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};
int main()
{
    enum week day;
    day = Wed;
    printf("%d",day);
    return 0;
}
```

Enum [2]

- E você pode definir os valores do enum
  - Mas será sequencial, começando em 0, caso contrário

```c
enum State {Working = 1, Failed = 0, Freezed = 0};

int main()
{
   printf("%d, %d, %d", Working, Failed, Freezed);
   return 0;
}
```

Rand() [3]

- Função da biblioteca *stdlib.h*
- Retorna um número pseudo-aleatório
- Usando o módulo é possível definir o intervalo do número criado:

```
v1 = rand() % 100;          // v1 in the range 0 to 99
v2 = rand() % 100 + 1;      // v2 in the range 1 to 100
v3 = rand() % 30 + 1985;    // v3 in the range 1985-2014
```

Voltando ao Exemplo de um Simulador de Dano

```c
enum types{Fire, Water, Wind, Earth, Light, Dark}; //0 a 5
int main() {
    int defenderHP, attackerAtk, defenderType, attackerType, damageMultiplier, hitChance, randHit;
    scanf("%d %d %d %d %d", &defenderHP, &defenderType, &attackerAtk, &attackerType, &hitChance);
    randHit = rand()%100;
    if(randHit < hitChance) {
        if(attackerType == Water && defenderType == Fire)
            damageMultiplier = 2;
        else if(attackerType == Wind && defenderType == Earth)
            damageMultiplier = 2;
        else if(attackerType == Light && defenderType == Dark)
            damageMultiplier = 2;
        else if(attackerType == Fire && defenderType == Water)
            damageMultiplier = 0.5;
        else if(attackerType == Earth && defenderType == Wind)
            damageMultiplier = 0.5;
        else if(attackerType == Dark && defenderType == Light)
            damageMultiplier = 0.5;
        else
            damageMultiplier = 1.0;
        defenderHP -= attackerAtk*damageMultiplier;
    }
    else
        printf("O ataque falhou!\n");
    return 0;
}
```

Exemplo de um Simulador de Dano V2

```c
enum types{Fire, Water, Wind, Earth, Light, Dark}; //0 a 5
int main(){
    int defenderHP, attackerAtk, defenderType, attackerType, damageMultiplier, hitChance, randHit;
    scanf("%d %d %d %d %d", &defenderHP, &defenderType, &attackerAtk, &attackerType, &hitChance);
    randHit = rand()%100;
    if(randHit < hitChance){
        if((attackerType == Water && defenderType == Fire) || (attackerType == Wind && defenderType == Earth)
          || (attackerType == Light && defenderType == Dark))
            damageMultiplier = 2;
        else if((attackerType == Fire && defenderType == Water) || (attackerType == Earth && defenderType ==
          Wind) || (attackerType == Dark && defenderType == Light))
            damageMultiplier = 0.5;
        else
            damageMultiplier = 1.0;
        defenderHP -= attackerAtk*damageMultiplier;
    }
    else
        printf("O ataque falhou!\n");
    return 0;
}
```

X Evitem isso! X

```c
enum types{Fire, Water, Wind, Earth, Light, Dark}; //0 a 5

int main()
{
    int defenderHP, attackerAtk, defenderType,
attackerType;
    int damageMultiplier, hitChance, randHit;
    scanf("%d %d", &defenderHP, &defenderType);
    scanf("%d %d %d", &attackerAtk, &attackerType,
&hitChance);
    randHit = rand()%100;
    if(randHit < hitChance)
    {
        if(attackerType == Water)
            if(defenderType == Fire)
                damageMultiplier = 2;
            else
                damageMultiplier = 1;
        else if(attackerType == Wind)
            if(defenderType == Earth)
                damageMultiplier = 2;
            else
                damageMultiplier = 1;
        else if(attackerType == Light)
            if(defenderType == Dark)
                damageMultiplier = 2;
            else
                damageMultiplier = 1;

                    • • •
```

```c
                                • • •
else if(attackerType == Fire)
            if(defenderType == Water)
                damageMultiplier = 0.5;
            else
                damageMultiplier = 1;
        else if(attackerType == Earth)
            if(defenderType == Wind)
                damageMultiplier = 0.5;
            else
                damageMultiplier = 1;
        else if(attackerType == Dark)
            if(defenderType == Light)
                damageMultiplier = 0.5;
            else
                damageMultiplier = 1;
        else
        {
            damageMultiplier = 1.0;
        }
        defenderHP -= attackerAtk*damageMultiplier;
    }
    else
    {
        printf("O ataque falhou!\n");
    }
    return 0;
}
```

```php
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```

Evitem isso também!

Fonte: http://i.imgur.com/BtjZedW.jpg

Evitando os "*ifs Hadouken*"

- Normalmente, se você está com muitos *ifs* aninhados, provavelmente vai ser muito mais fácil inverter a condição e tratar os casos negativos *(elses*)
- Outras vezes funções podem te ajudar. Estudaremos sobre elas mais adiante no curso :)

Switch-Case

Switch-Case

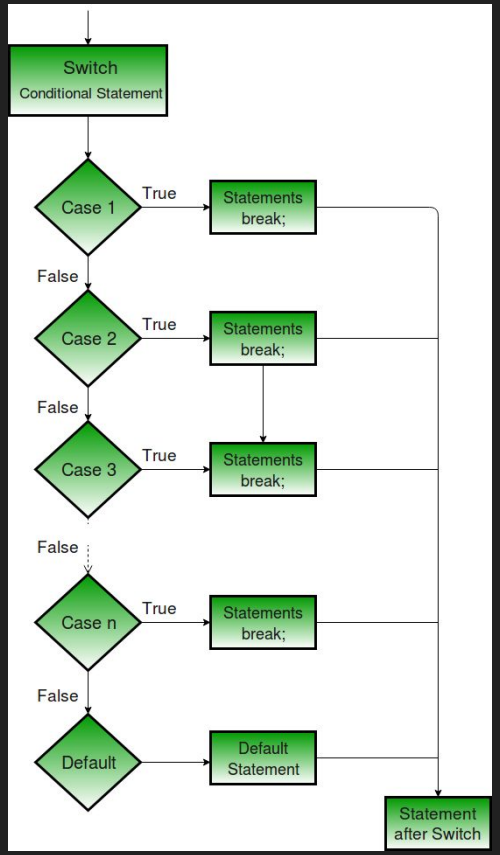
- Outro tipo de operador condicional
- Especialmente recomendado para substituir um if quando uma variável é comparada a um inteiro várias vezes
- Muito recomendado para escolhas de menu em que as opções são números

Switch-Case [4]

```
switch (n)
{
    case 1: // code to be executed if n = 1;
        break;
    case 2: // code to be executed if n = 2;
        break;
    default: // code to be executed if n doesn't match any
cases
}
```

Switch-Case [4]

- O comando *break* é usado para retornar o controle de execução do código para o primeiro comando depois de um *loop* ou uma sequência de declarações no *switch*
- Caso ele não esteja no código, o próximo caso será testado, até encontrar um *break*.
- Você também pode aninhar *Switch-Cases*, mas dificuldade a legibilidade.



Fonte: https://www.geeksforgeeks.org/switch-statement-cc/

Switch-Case [5]

- A expressão precisa ser um tipo inteiro (int, char, enum)
  - Um float, por exemplo, não pode ser usado
- A expressão usada nas *labels* (após o *case*) precisam ser expressões constantes
  - Não podem ser variáveis
- Cada *label* precisa ter um valor único!

Switch-Case

```c
int main(){
    int option, a=1;
    scanf("%d", &option);
    switch (option)
    {
        case 1: printf("Escolheu a 1\n");
            break;
        //Não pode ser uma variável
        //case a: printf("Escolheu a 2\n");
        case 2: printf("Escolheu a 2\n");
            break;
        case 3: printf("Escolheu a 3\n");
            break;
        default: printf("Não existe essa opção\n");
            //Não precisa de break já que é o fim. Mas pode colocar
            //break;
    }
    return 0;
}
```

Switch-Case

```c
int main(){
    char optionChar;
    scanf("%c", &optionChar);
    switch (optionChar)
    {
        case '1': printf("Escolheu a char 1\n");
            break;
        case '2': printf("Escolheu a char 2\n");
            break;
        case 'a': printf("Escolheu a char a\n");
            break;
        default: printf("Não existe essa opção\n");
            break;
    }
    return 0;
}
```

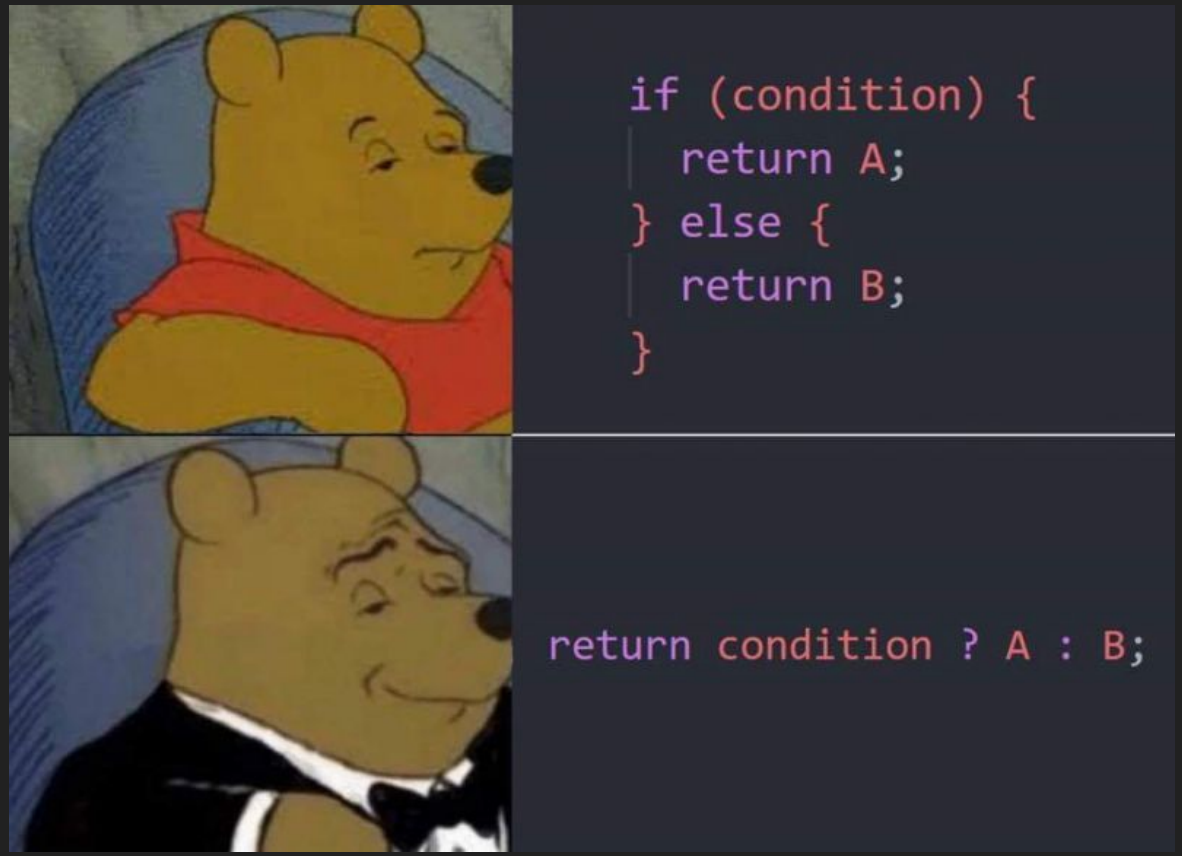
Switch-Case

- Existe a noção que *Switch-Case* é sempre mais rápido que um *If-Else*
- Isso nem sempre é verdade!
- No caso do C (e em outros compiladores mais antigos) ele cria uma *jump table* (ou *branch table*) quando o *Switch-case* tem 5 ou mais casos
- É uma estrutura que otimiza o acesso ao caso desejado, uma vez que ela não processa linearmente as opções

Switch-Case

- Para esses casos, o acesso realmente é mais rápido.
- Porém, em linguagens mais recentes, boa parte dos compiladores também faz o mesmo com o *If-Else* sempre que possível
- Em resumo:
  - Para o C, quando você tiver 5 ou mais opções de valores inteiros (preferencialmente sequenciais), *switch-case* é, sim, mais rápido.

Operador Ternário



Fonte: https://devrant.com/rants/2048071/pooh-memes-are-the-craze-now

Operador Ternário

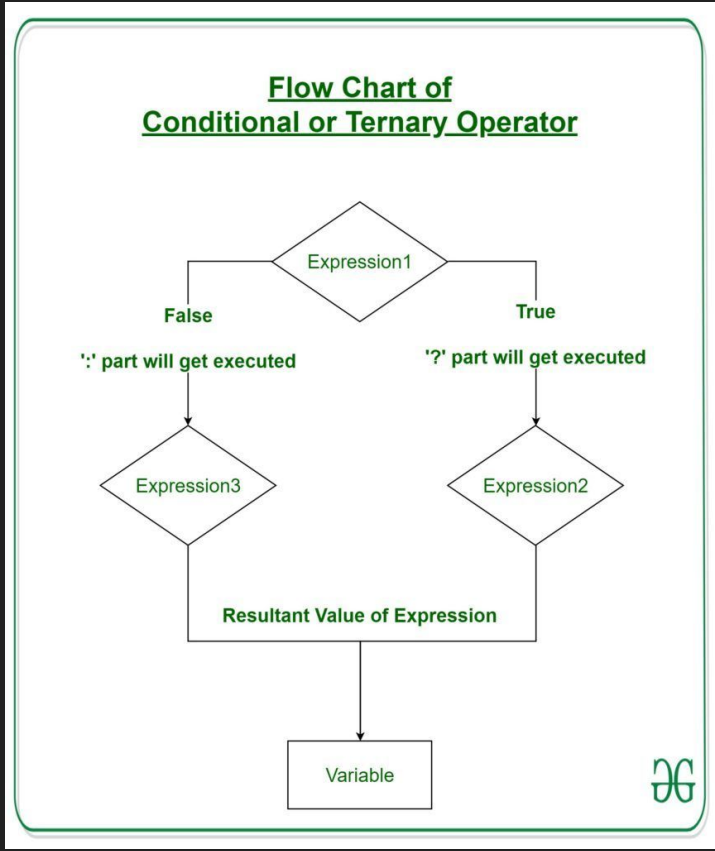
- É um jeito diferente de escrever estruturas *If-Else*
- Reduz bastante o tamanho da expressão
- Porém, dificulta a leitura quando aninhado ou muito extenso
- Sua sintaxe consiste em
  - *Variável* = *Expressão1* ? *Expressão2* : *Expressão2*

Operador Ternário

```
variable = Expression1 ? Expression2 : Expression3
```

➔ É equivalente a:

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



Fonte: https://www.geeksforgeeks.org/conditional-or-ternary-operator-in-c-c/

Operador Ternário

```c
int main()
{
    int n1 = 5, n2 = 10, max, min;
    max = (n1 > n2) ? n1 : n2;
    min = (n1 < n2) ? n1 : n2;
    //Equivalente a:
    //min = (n1 > n2) ? n2 : n1;
    printf("Max: %d, Min: %d\n", max, min);
    return 0;
}
```

Operador Ternário Aninhado

```c
int main()
{
    int n1 = 5, n2 = 10, max, min;
    int n3 = 8;
    //max = (n1>n2) ? (n1>n3) ? n1 : n3 : n2;
    max = (n1>n2)
        ? (n1>n3)
            ? n1
            : n3
        : n2;
    printf("Max out of three: %d", max);
}
```

Operador Ternário Aninhado

```c
int main()
{
    int n1 = 5, n2 = 10, n3 = 8, max, min, middle;
 // middle = (n1>n2) ? (n1>n3) ? (n2>n3) ? n2 : n3 : n1 : (n2>n3) ? (n1>n3) ? n1 : n3 : n2;
    middle = (n1>n2)
    ? (n1>n3)
        ? (n2>n3)
            ? n2
            : n3
        : n1
    : (n2>n3)
        ? (n1>n3)
            ? n1
            : n3
        : n2;
    printf("Middle out of three: %d", middle);
}
```

Comparação de Ponto Flutuante

Comparação de ponto flutuante

- Devido aos erros de arredondamento de ponto flutuante, não é recomendado usar uma comparação de “==” para pontos flutuantes.
- Uma solução é verificar se a diferença entre eles é menor do que um valor arbitrariamente pequeno.

Comparação Ponto Flutuante

```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
    float a = 0.1, b = 0.1;
    if(a==0.1f)
        printf("Igualei dois floats\n");
    else
        printf("Não consegui igualar dois floats\n");
    if(a==0.1)
        printf("Igualei dois float e double\n");
    else
        printf("Não consegui igualar float e double\n");
    if(a==b)
        printf("Igualei dois floats\n");
    else
        printf("Não consegui igualar dois floats\n");
    a += 1.2;
    b += 1.2f;
    if(a==b)
        printf("Igualei dois floats depois de um cast implicito\n");
    else
        printf("Não consegui dois floats depois de um cast implicito\n");
    if(abs(a-b) < 0.0001)
        printf("Agora igualei dois floats depois de um cast implicito\n");
}
```

Referências

[1] https://www.geeksforgeeks.org/decision-making-c-c-else-nested-else/

[2] https://www.geeksforgeeks.org/enumeration-enum-c/

[3] http://www.cplusplus.com/reference/cstdlib/rand/

[4] https://www.geeksforgeeks.org/switch-statement-cc/

[5] https://www.geeksforgeeks.org/interesting-facts-about-switch-statement-in-c/

[6] https://www.geeksforgeeks.org/break-statement-cc/

[7] https://www.geeksforgeeks.org/conditional-or-ternary-operator-in-c-c/

[8] https://www.geeksforgeeks.org/c-nested-ternary-operator/

[9] https://www.geeksforgeeks.org/comparison-float-value-c/