

Programação 1

Aula 2

Valeri Skliarov, Prof. Catedrático

Email: skl@ua.pt

URL: <http://sweet.ua.pt/skl/>

Departamento de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro

<http://elearning.ua.pt/>

Pontos importantes

```
import java.util.*;
```

Importar a biblioteca de classes (Scanner)

```
Scanner sc = new Scanner(System.in);
```

Criar um objeto novo do tipo Scanner

Função nextInt do objeto do tipo Scanner

```
a = sc.nextInt();
```

Utilizar o objeto do tipo Scanner

Exemplo 1:

1. Importar Scanner

```
import java.util.*;
```

2. Criar objeto

```
Scanner objeto = new Scanner(System.in);
```

3. Usar objeto

```
int inteiro;
```

```
double real_v;
```

```
System.out.print("Inteiro:");
```

```
inteiro = objeto.nextInt();
```

```
System.out.print("Real:");
```

```
real_v = objeto.nextDouble();
```

```
System.out.println("inteiro é " + inteiro + "; real é " + real);
```

```
System.out.printf("inteiro é %d; real é %f\n", inteiro, real);
```

```
objeto.close();
```

4. Fechar objeto

```
}  
}
```

```
Inteiro:2  
Real:34343  
inteiro e 2; real e 34343.0  
inteiro e 2; real e 34343.000000
```

Exemplo 2:

Erro

```
import java.util.*;
```

```
public class Nome {  
    public static void main(String[] args) {  
        Scanner objeto = new Scanner(System.in);  
        int inteiro;  
        double real;  
        System.out.print("Inteiro:");  
        inteiro = objeto.nextInt();  
        real = Ler_real();  
    }  
}
```

```
System.out.println("inteiro é " + inteiro + "; real é " + real);  
System.out.printf("inteiro é %d; real é %f\n",inteiro,real);  
}
```

```
public static double Ler_real()  
{  
    double real;  
    System.out.print("Real:");  
    real = objeto.nextDouble();  
    return real;  
}  
}
```

Exemplo 3:

Ok

```
import java.util.*;
```

```
public class Nome {  
    public static void main(String[] args) {  
        Scanner objeto = new Scanner(System.in);
```

```
        int inteiro;
```

```
        double real;
```

```
        System.out.print("Inteiro:");
```

```
        inteiro = objeto.nextInt();
```

```
        real = Ler_real();
```

```
        System.out.println("inteiro é " + inteiro + "; real é " + real);
```

```
        System.out.printf("inteiro é %d; real é %f\n",inteiro,real);
```

```
        objeto.close();
```

```
    }
```

```
    public static double Ler_real()
```

```
    {
```

```
        Scanner objeto = new Scanner(System.in);
```

```
        double real;
```

```
        System.out.print("Real:");
```

```
        real = objeto.nextDouble();
```

```
        objeto.close();
```

```
        return real;
```

```
    }
```

```
}
```

Identificadores

Identificadores – nomes utilizados para designar todos os objetos existentes num programa. Devem começar por uma letra ou por símbolo ‘_’ e só podem conter letras, números e símbolo ‘_’ (ex. nome, idade, i, j, cont_1, dia_mes, _km, res, nome1, n1_and_n3, a_____123_____b, t_, ...).

Exemplos errados: 3i, 1__nome, 22, meu nome, o maior

Espaço não pode ser usado

Estilo recomendado

- Para nomear itens no seu programa **não pode** usar nomes reservados em Java (**class**, **new**, **int**, **public**, etc.).
- Aconselha-se que:
 - os **nomes das classes** comecem sempre por letra maiúscula; se o nome da classe inclui várias palavras, escreva cada uma destas com letra maiúscula: **Pessoa**, **ContaBancaria**;
 - todas as entidades restantes (**variáveis**, métodos, etc.) só incluam letras minúsculas exceto se se tratar de palavras compostas em que deve escrever cada uma destas palavras com letra maiúscula exceto a primeira: **nomeCompleto**, **capacidadeDeposito**, **ano**, **calcularSaldo**.;
 - para variáveis sejam usados nomes descritivos (p.ex. **dia**, **mes**, **dist**, em vez de **a**, **b**, **c**).

Alcance de variáveis

Um alcance é delimitado por um par de chavetas {}.

A variável definida num alcance só existe dentro deste alcance. Quando se sai deste a variável é destruída.

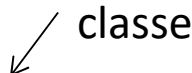
Exemplo:

```
{  
    int x = 12;  
    // aqui a variável x está disponível  
    {  
        int q = 96;  
        //int x = 2; não se pode declarar mais um x  
        // aqui x e q estão disponíveis  
    }  
    // aqui apenas x está disponível  
} // nem x, nem q estão disponíveis
```

The diagram uses vertical arrows to show the scope of variables. A long arrow on the left, labeled 'alcance de x', spans from the opening curly brace of the outer block to the closing curly brace, indicating that variable 'x' is only available within this outer block. A shorter arrow inside, labeled 'alcance de q', spans from the opening curly brace of the inner block to its closing curly brace, indicating that variable 'q' is only available within the inner block.

- Estruturas de controlo – decisão
- Tipos de dados **boolean**
- Operadores relacionais
- Operadores lógicos
- Estrutura de decisão **if**
- Estrutura de decisão múltipla **switch**

Alguns conceitos essenciais...

- Tipo de dados **boolean** (ou **Boolean**) – podem assumir valores **true** e **false** (verdadeiro e falso).

- Operadores relacionais: `<`, `<=`, `>`, `>=`, `==`, `!=`
- Operadores lógicos: `!`, `||`, `&&`
- Exemplos:

```
boolean cond1, cond2, cond3, cond4, cond5;
cond1 = 3 > 0;           // cond1 fica com true
cond2 = 5 != 5;          // cond2 fica com false
cond3 = cond1 || cond2;  // true || false cond3
                        // fica com true
cond4 = cond1 && cond2;    // true && false cond4
                        // fica com false
cond5 = !cond4;          // !false cond5 fica com true
```

Operadores - prioridade

- A ordem de execução de operadores numa expressão complexa rege-se pelas **regras de precedência**.

```
int a = 5;
```

```
int b = -5;
```

```
int c = ++a&b>>>30;
```

- Para alterar a ordem e/ou clarificar as expressões complexas sugere-se que usem parênteses.

```
c = (++a) & (b>>>30);
```

Operators	Precedence
postfix	expr++ expr--
unary	++expr --expr +expr -expr ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	
logical AND	&&
logical OR	
ternary	? :
assignment	= += -= *= /= %= &= ^= = <<= >>= >>>=

Estruturas de controlo - decisão

- Uma das particularidades de um computador é a capacidade de repetir tarefas ou executar tarefas consoante determinadas condições.
- Para implementar programas mais complexos, temos a necessidade de executar instruções de forma condicional.
- Determinadas instruções só podem/devem ser executadas depois da avaliação de determinadas condições.
- As instruções que permitem condicionar a execução de outras designam-se por *estruturas de controlo*. Nestes slides vamos apresentar as *estruturas de decisão*.
- Temos em JAVA dois tipos de instruções de decisão: **if** e **switch**.

Instrução de decisão `if`

- `if` (expressão) instrução;
- a expressão é avaliada;
- tem que ser uma expressão cujo resultado seja do tipo booleano;
- se verdadeira, é executada a instrução;
- se falsa, o programa continua na linha seguinte;
- exemplo:

```
int x;  
System.out.print("Um valor inteiro:");  
x = sc.nextInt();  
if( x < 0)  
    x = -x;  
System.out.println("O valor absoluto é " + x);
```

Instrução de decisão **if**

```
if (expressão) { bloco1; }  
else           { bloco2; }
```

- a expressão é avaliada;
- se verdadeira, é executado o bloco1;
- se falsa, é executado o bloco2;
- se um bloco contiver mais de uma instrução, deve levar chavetas.

Exemplo:

```
if (item1 >= item2)  
    System.out.println("primeiro é maior ou igual a segundo");  
else  
    System.out.println("segundo é o maior");
```

Resultados

```
import java.util.*;
public class Igualdade
{
    public static void main (String args[])
    {
        Scanner sc = new Scanner(System.in);
        int A, B;
        boolean a, b;
        A = sc.nextInt(); B = sc.nextInt();
        if (A == B)
            System.out.println("A = B");
        else
            System.out.println("A != B");
        if (A != B)
            System.out.println("A != B");
        else
            System.out.println("A = B");
        System.out.println(A == B ? "A = B" : "A != B");
    }
}
```

```
3
6
A != B
A != B
A != B
Press any key to continue . . .
```

```

import java.util.*;
public class Igualdade4 {
    public static void main (String args[])
    {
        Scanner sc = new Scanner(System.in);
        int A, B, C, D;
        boolean y, a, b, c, d, e, f;
        A = sc.nextInt(); B = sc.nextInt(); C = sc.nextInt(); D = sc.nextInt();
        a = A == B; // a é true se A for igual a B e a é false no caso opósito
        b = B == C;
        c = C == D;
        d = A == C;
        e = A == D;
        f = B == D;
        if (a && b && c)
            System.out.println("A = B = C = D");
        y = !a && !b && !c && !d && !e && !f;
        if (y)
            System.out.println("todos os valores A, B, C, D são diferentes");
        if (a || b || c || d || e || f)
            System.out.println("pelo menos dois valores do conjunto {A, B, C, D} sao iguais");
        System.out.printf("a and b and c = %b\n", a && b && c);
        System.out.printf("not(not a or not b or not c) = %b\n", !(a || b || c));
    }
}

```

Resultados

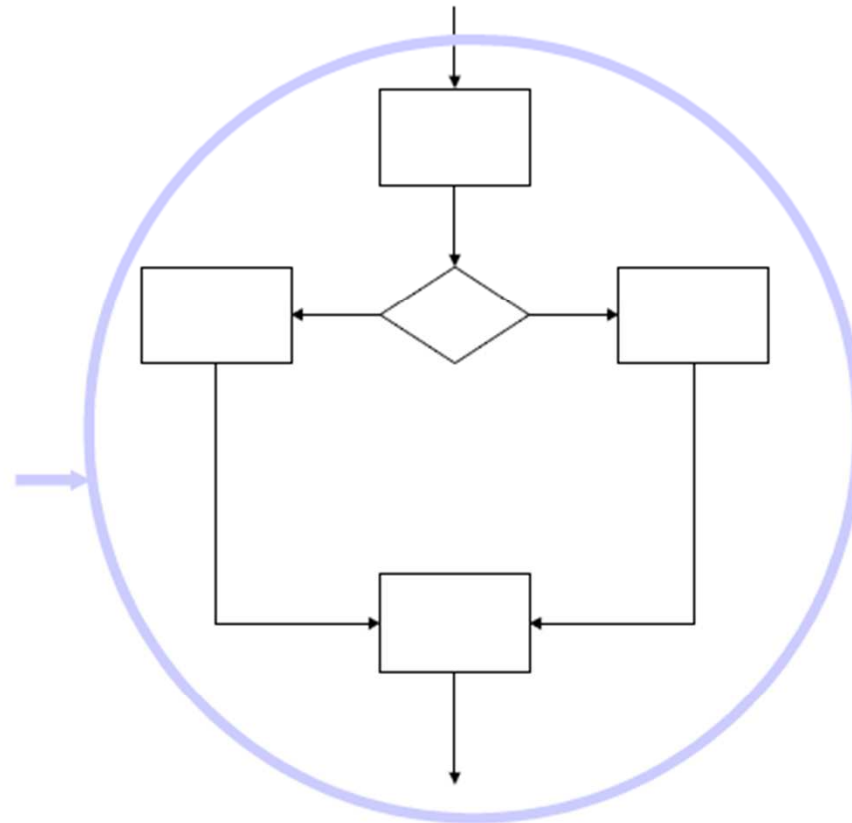
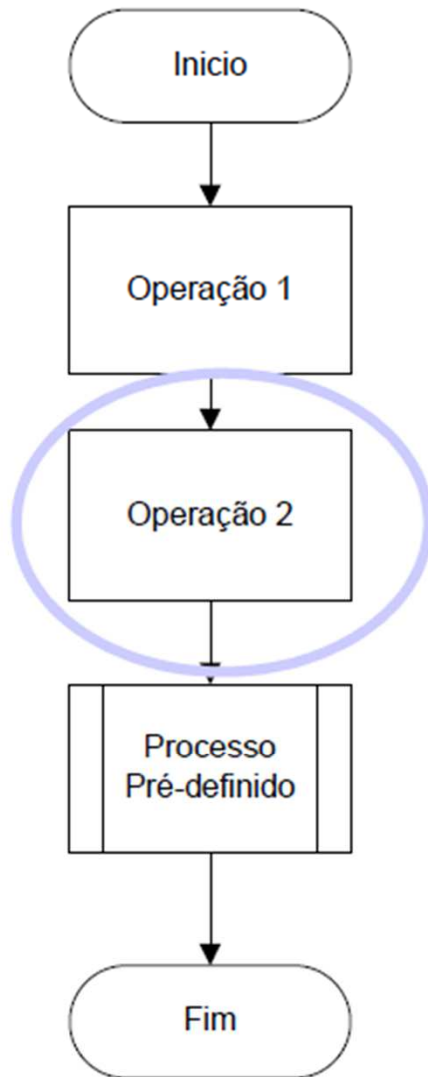
```

1
2
1
5
pelo menos dois valores do conjunto {A, B, C, D} sao iguais
a and b and c = false
not(not a or not b or not c) = false
Press any key to continue . . . _

```

Para imprimir um valor booleano

Diagramas de fluxo (*flowcharts*)



Exercício com instrução **if**

Verificar a paridade dum número inteiro.

```
Scanner sc = new Scanner(System.in);  
int number = sc.nextInt();  
if (number % 2 == 0)  
    System.out.println("O número é par");  
else  
    System.out.println("O número é ímpar");  
sc.close();
```

Instrução de decisão **if**

- A seguir à instrução decisória `if` ou ao separador `else`, podemos ter qualquer tipo de instrução, inclusive outras instruções de decisão.

```
if(condiçao1)
{
    if(condição2)
    {
        bloco1;
    }
    else
    {
        bloco2;
    }
}
else
{
    bloco3;
}
```

```
if(condiçao1)
{
    bloco1;
}
else if(condição2)
{
    bloco2;
}
else
{
    bloco3;
}
```

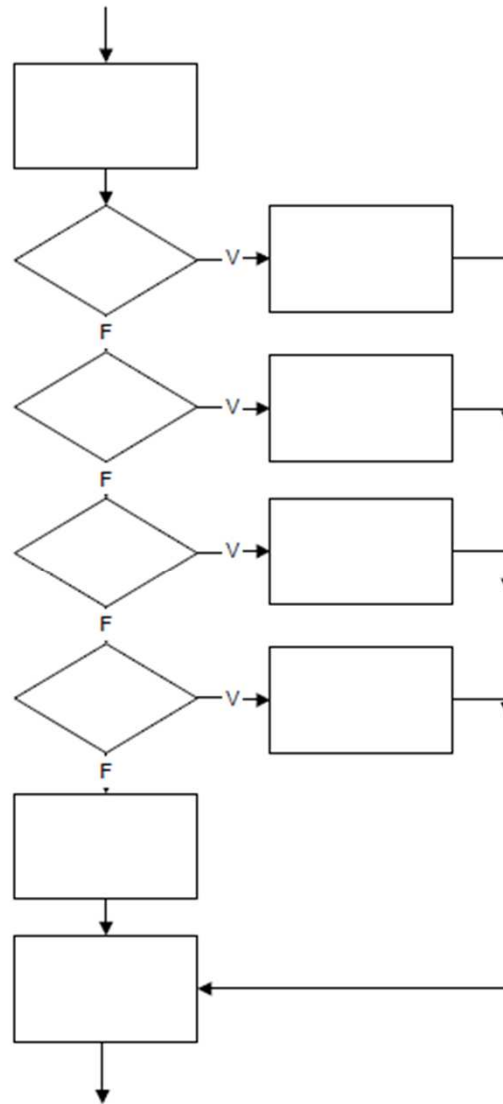
Instrução de decisão múltipla `switch`

Algumas situações de decisão encadeadas com a instrução `if` podem ser resolvidas através da instrução de decisão múltipla **`switch`**.

- **`switch`** (expressão)
- {
- **`case`** valor1:
- bloco1;
- **`break`**;
- **`case`** valor2:
- bloco2;
- **`break`**;
- **`default`**:
- bloco3;
- }

- A expressão deve ser do tipo enumerado (número inteiro ou carater no caso dos tipos primitivos de JAVA – **`byte`, `short`, `int` ou `char`**).
- As constantes que constituem a lista de alternativas são do mesmo tipo da expressão.
- Primeiro é calculada a expressão e depois o seu valor é pesquisado na lista de alternativas existentes em cada **`case`**, pela ordem com que são especificados.
- Se a pesquisa for bem sucedida, o bloco de código correspondente é executado.
- Caso não exista na lista e se o **`default`** existir, o bloco de código correspondente é executado.
- A execução do **`switch`** só termina com o aparecimento da instrução **`break`**.

Diagramas de fluxo (*flowcharts*)



Exemplo 1:

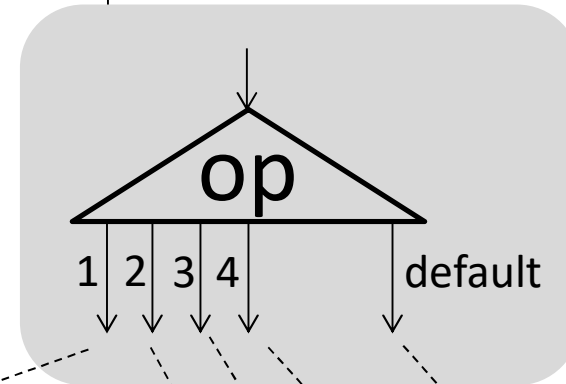
```
import java.util.*;

public class ex_switch
{
    public static void main (String args[])
    {
        Scanner sc = new Scanner(System.in);

        int a=2, b=3;
        int op;

        System.out.print("op ?");
        op = sc.nextInt();

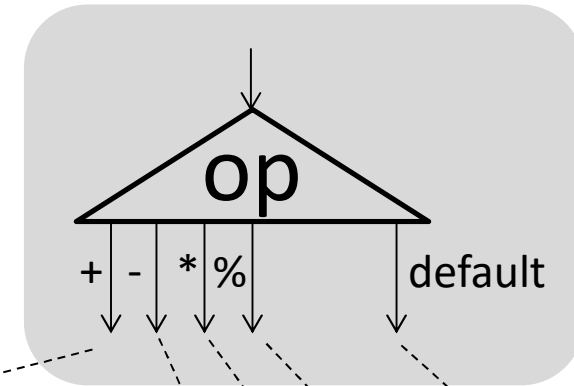
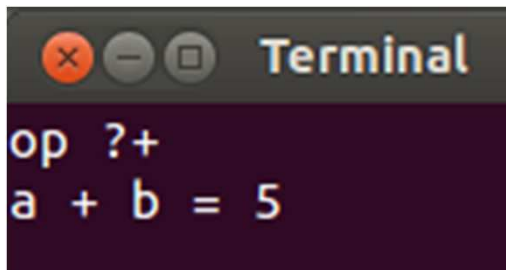
        switch(op)
        {
            case 1: <
                System.out.printf("a + b = %d\n", a+b);
                break; // remover break e experimentar
            case 2:
                System.out.printf("a - b = %d\n", a-b); <
                break;
            case 3:
                System.out.printf("a * b = %d\n", a*b); <
                break;
            case 4:
                System.out.printf("a %% b = %d\n", a%b); <
                break;
            default:
                System.out.println("Wrong operation"); <
        }
    }
}
```



A terminal window titled 'Terminal' showing the execution of the program. The prompt 'op ?' is followed by the input '3'. The output is 'a * b = 6'.

Exemplo 2:

```
import java.util.*;
public class ex_switch_char
{
    public static void main (String args[])
    {
        Scanner sc = new Scanner(System.in);
        int a=2, b=3;  char op;
        System.out.print("op ?");
        op = sc.nextLine().charAt(0);
        switch(op)
        {
            case '+':
                System.out.printf("a + b = %d\n", a+b);
                break;
            case '-':
                System.out.printf("a - b = %d\n", a-b);
                break;
            case '*':
                System.out.printf("a * b = %d\n", a*b);
                break;
            case '%':
                System.out.printf("a %% b = %d\n", a%b);
                break;
            default:
                System.out.println("Operação errada");
        }
    }
}
```



Switch sem break

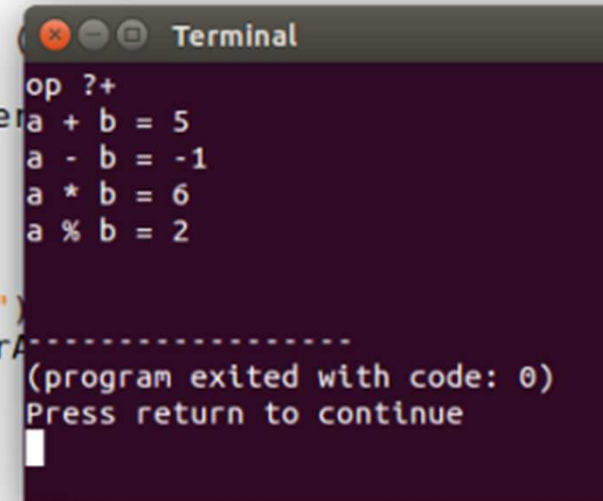
```
import java.util.*;

public class ex_switch_char
{
    public static void main (
    {
        Scanner sc = new Scanner

        int a=2, b=3;
        char op;

        System.out.print("op ?")
        op = sc.nextLine().charA

        switch(op)
        {
            case '+':
                System.out.printf("a + b = %d\n", a+b);
                // break;
            case '-':
                System.out.printf("a - b = %d\n", a-b);
                // break;
            case '*':
                System.out.printf("a * b = %d\n", a*b);
                // break;
            case '%':
                System.out.printf("a %% b = %d\n", a%b);
                break;
            default:
                System.out.println("Operacao errada");
        }
    }
}
```



```
Terminal
op ?+
a + b = 5
a - b = -1
a * b = 6
a % b = 2
-----
(program exited with code: 0)
Press return to continue
```


Exercício com instrução switch

Determinar se uma letra é vogal.

```
public class Vogais
{
    public static void main(String[] args) {
        Scanner ler = new Scanner(System.in);
        char letra;
        System.out.println("Introduza uma letra:");
        letra = ler.nextLine().charAt(0);
        switch (letra) {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':
                System.out.printf("Vogal %c\n", letra);
                break;
            default:
                System.out.printf("Nao e vogal. O codigo do %c = %d\n", letra, (byte)letra);
        }
    }
}
```

Código ASCII

Diagram illustrating the ASCII code mapping across three columns: Código binário (Binary), Código octal (Octal), Código decimal (Decimal), and Código hexadecimal (Hexadecimal).

Arrows indicate the mapping from the binary code to the octal, decimal, and hexadecimal codes.

Código binário	Código octal	Código decimal	Código hexadecimal
010 0001	041	33	21
010 0010	042	34	22
010 0011	043	35	23
010 0100	044	36	24
010 0101	045	37	25
010 0110	046	38	26
010 0111	047	39	27
010 1000	050	40	28
010 1001	051	41	29
010 1010	052	42	2A
010 1011	053	43	2B
010 1100	054	44	2C
010 1101	055	45	2D
010 1110	056	46	2E
010 1111	057	47	2F
011 0000	060	48	30
011 0001	061	49	31
011 0010	062	50	32
011 0011	063	51	33
011 0100	064	52	34
011 0101	065	53	35
011 0110	066	54	36
011 0111	067	55	37
011 1000	070	56	38
011 1001	071	57	39
011 1010	072	58	3A
100 0001	101	65	41
100 0010	102	66	42
100 0011	103	67	43
100 0100	104	68	44
100 0101	105	69	45
100 0110	106	70	46
100 0111	107	71	47
100 1000	110	72	48
100 1001	111	73	49
100 1010	112	74	4A
100 1011	113	75	4B
100 1100	114	76	4C
100 1101	115	77	4D
100 1110	116	78	4E
100 1111	117	79	4F
101 0000	120	80	50
101 0001	121	81	51
101 0010	122	82	52
101 0011	123	83	53
101 0100	124	84	54
101 0101	125	85	55
101 0110	126	86	56
101 0111	127	87	57
101 1000	130	88	58
101 1001	131	89	59
101 1010	132	90	5A
110 0001	141	97	61
110 0010	142	98	62
110 0011	143	99	63
110 0100	144	100	64
110 0101	145	101	65
110 0110	146	102	66
110 0111	147	103	67
110 1000	150	104	68
110 1001	151	105	69
110 1010	152	106	6A
110 1011	153	107	6B
110 1100	154	108	6C
110 1101	155	109	6D
110 1110	156	110	6E
110 1111	157	111	6F
111 0000	160	112	70
111 0001	161	113	71
111 0010	162	114	72
111 0011	163	115	73
111 0100	164	116	74
111 0101	165	117	75
111 0110	166	118	76
111 0111	167	119	77
111 1000	170	120	78
111 1001	171	121	79
111 1010	172	122	7A