

ELAS.NET APRESENTA

Curso Internet das Coisas

Respostas dos Exercícios da Aula de Revisão



Índice

Variáveis e Tipos de dados básicos	02
Funções	04
Operadores	05
Comandos de Decisão	07
Comandos de Repetição	08
Considerações Finais	12



Pergunta: Variáveis e Tipos de dados básicos

Qual das alternativas apresenta a forma correta de se declarar e inicializar uma variável do tipo inteiro?

- ☐ A `soma = 16`
- ☐ B `float soma = 16`
- ☐ C `int soma = 16;`
- ☐ D `int soma = 16`



Resposta: C

Explicação

Sintaxe: `tipo nomeVariavel = valor;`

Letra A está errada, pois a variável não está sendo declarada (trecho de declaração de variável: `tipo nomeVariavel`) e a linha não é finalizada com ponto e vírgula(;).

Letra B está errada pois a palavra chave para valores inteiros é `int` e a linha não é finalizada com ponto e vírgula(;).

Letra C está correta pois segue a sintaxe.

Letra D está errada, pois a linha não é finalizada com ponto e vírgula(;).

Pergunta: Variáveis e Tipos de dados básicos

Qual das alternativas apresenta a forma correta de se declarar e inicializar uma variável do tipo caracter?

- (A) `char = 'a'`
- (B) `char letra = 'a';`
- (C) `char letra = a;`
- (D) `char letra = "a";`
- (E) `char letra = 'ab';`



Resposta: B

Explicação

Sintaxe: `tipo nomeVariavel = valor;`

Importante: Valores char devem estar entre aspas simples ("").

Letra A está errada, pois é preciso especificar o identificador (nome) da variável e a linha não é finalizada com ponto e vírgula(;;).

Letra B está correta, pois segue a sintaxe.

Letra C está errada, pois não inicializa o valor entre aspas simples.

Letra D está errada, pois inicializa o valor entre aspas duplas (isso só estaria correto se a variável representasse uma string - cadeia de caracteres).

Letra E está errada, pois inicializa o valor de forma incorreta. (Forma correta - declarando a variável e inicializando com string: `char letra[3] = "ab";`)

Pergunta: Funções

Qual das alternativas apresenta a forma correta de chamar a função abaixo e o valor retornado?

```
int geraIdade(int anoNascimento, int anoAtual){  
    return anoAtual-anoNascimento;  
}
```

- (A) idade = geraIdade(2004,2021); //retorno: 17
- (B) idade = geraIdade(2021,2004); //retorno: 17
- (C) idade = geraIdade(); //não retorna nenhum valor
- (D) idade = geraIdade(2021,2004); //não retorna nenhum valor



Resposta: A

Explicação

Letra A está correta, pois, ao passar os argumentos na chamada do método, `anoNascimento` recebe 2004 e `anoAtual` recebe 2021. Então, o valor retornado será $2021 - 2004 = 17$.

Letra B está errada, pois, ao passar os argumentos na chamada do método, `anoNascimento` recebe 2021 e `anoAtual` recebe 2004. Então, o valor retornado será $2004 - 2021 = -17$.

Letra C está errada, pois não passa nenhum argumento na chamada do método - que em sua declaração possui dois parâmetros do tipo inteiro. Além disso, a função `geraIdade` possui retorno do tipo inteiro. entre aspas simples.

Letra D está errada, a função `geraIdade` possui retorno do tipo inteiro e neste caso é -17.

Pergunta: Operadores

Qual das alternativas apresenta corretamente o valor retornado pelas expressões?

- (A) `10+5 > 3*5 && 'a' == 'a' //valor retornado: true`
- (B) `!('Ana' == 'ana') //valor retornado: false`
- (C) `5 % 2 == 1 && (9-3) >= 6 //valor retornado: false`
- (D) `1+1 == 2 || 'A' == 'a' //valor retornado: false`



Resposta: D

Explicação

Nesta questão vamos analisar cada parte das expressões.

Letra A está errada, pois:

`10+5 > 3*5 && 'a' == 'a'`

`10+5 > 3*5` → `false`, afinal 15 é igual a 15 e não maior.

`'a' == 'a'` → `true`, afinal 'a' é igual a 'a' e não maior.

`false && true` → `false`

Letra B está errada, pois:

`!("Ana" == "ana")`

`"Ana" == "ana"` → `false`, lembre-se que letras maiúsculas e minúsculas são caracteres diferentes.

`!false` → `true`

Letra C está errada, pois:

`5 % 2 == 1 && (9-3) >= 6`

`5 % 2 == 1` → `true`, afinal o resto da divisão de 5 por 2 é igual a 1.

`(9-3) >= 6` → `true`, afinal 9-3 é maior ou igual a 6.

`true && true` → `true`

Letra D está correta, pois:

```
1+1 == 2 || 'A' == 'a'
```

`1+1 == 2` → `true` , afinal `1+1` é igual a `2`.

`'A' == 'a'` → `false`, afinal letras maiúsculas e minúsculas são caracteres diferentes.

```
true || false → true
```

Pergunta: Comando de Decisão

Qual das alternativas apresenta corretamente o valor que a variável resultado assumirá ao final da execução?

```
if (!('Ana' == 'ana')){
    resultado = 0;
} else {
    resultado = 1;
}

if (resultado == 1){
    resultado += 2;
}

if (resultado > 0){
    resultado--;
}
```

- (A) 0
- (B) 1
- (C) 2
- (D) 3



Resposta: A

Explicação

Vamos analisar cada linha:

true → entra no if e resultado recebe o valor 0.

```
if (!("Ana" == "ana")){
    resultado = 0;
} else {
    resultado = 1;
}
//neste ponto, resultado = 0
```

false → não entra neste if

```
if (resultado == 1){
    resultado += 2;
}
//neste ponto, resultado = 0
```

false → não entra neste if

```
if (resultado > 0){
    resultado--;
}
//neste ponto, resultado = 0
```


Pergunta: Comando de Repetição

Qual das alternativas apresenta corretamente o valor que a variável soma assumirá ao final da execução?

```
int soma = 0;

for (i = 1; i <= 5; i+2){
    soma = soma + i
}
```

- (A) 0
- (B) 4
- (C) 9
- (D) 15



Resposta: C

Explicação

Sintaxe: `for (variável; condição; atualização){`
 <instrução>;
}

```
int soma = 0;
```

```
for (i = 1; i <= 5; i+2){
    soma = soma + i
}
```

contador i iniciando em 1; o laço não para enquanto o contador é menor ou igual a 5; e o contador aumenta de 2 em 2.

Primeira repetição:

```
soma=0
i = 1
```

Então, soma é igual a $0 + 1 = 1$

Segunda repetição:

```
soma = 1
i = 1 + 2 = 3
```

Então, soma é igual a $1 + 3 = 4$

Terceira repetição:

```
soma = 4
i = 3 + 2 = 5
```

Então, soma é igual a $4 + 5 = 9$

A partir daí, o valor do contador i não se aplica mais à condição estabelecida no laço for.

Pergunta: Comando de Repetição

Qual das alternativas apresenta corretamente a quantidade de vezes que o laço while repetirá até parar?

```
bool verificador = true;
int contador = 0;

while(verificador == true){
    contador++;

    if(contador > 5){
        verificador = false;
    }
}
```

- (A) 0
- (B) 1
- (C) 5
- (D) 6
- (E) O laço não para



Resposta: D

Explicação

Sintaxe: while (condição){
 <instrução>;
}

```
bool verificador = true;
int contador = 0;

while(verificador == true){
    contador++;

    if(contador > 5){
        verificador = false;
    }
}
```

Primeira repetição:

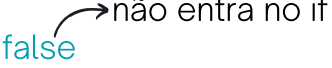
contador = 0 + 1 = 1

→ não entra no if
false

```
if(contador > 5){
    verificador = false;
}
```

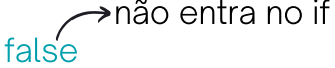
Segunda repetição:

contador = 1 + 1 = 2


`if(contador > 5){
 verificador = false;
}`

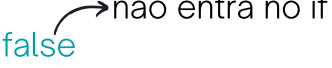
Terceira repetição:

`contador = 2 + 1 = 3`


`if(contador > 5){
 verificador = false;
}`

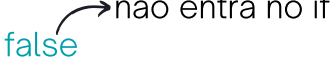
Quarta repetição:

`contador = 3 + 1 = 4`


`if(contador > 5){
 verificador = false;
}`


Quinta repetição:

`contador = 4 + 1 = 5`


`if(contador > 5){
 verificador = false;
}`

Sexta repetição:

`contador = 5 + 1 = 6`


`if(contador > 5){
 verificador = false;
}`

A partir daí, o valor de verificador não se aplica mais à condição estabelecida no laço while.

Considerações Finais

É importante que você entenda bem os exercícios propostos nessa aula assim como suas resoluções. Nos exercícios sobre Comando de Decisão e de Repetição, é preciso estar atento se as condições estão sendo atendidas para saber se o fluxo do código entra ou não no bloco destes comandos.

Além disso, quando for escrever um código usando o que aprendemos neste módulo, lembre-se de ter sua lógica bem desenvolvida. Se preciso, escreva passo a passo aquilo que deseja fazer a partir daquele código e depois escreva seu programa seguindo esse passo a passo.

Qualquer dúvida, entre em contato e busque em fontes confiáveis - no material/slides de aula e nas fontes indicadas em aula.

Programação é prática, dedicação e paciência! Não deixe de tentar. VOCÊ CONSEGUE!

