Revisando C

Total de pontos 5/6

Vamos revisar a lógica de Programação usando a lingugem C

0 de 0 pontos

Objetivos

- Recordar os principais elementos da linguagem C
- Associar as entradas e saídas das questões
- Aplicar a lógica de programação na liguagem C
- Comparar códigos de problemas nos enunciados
- Avaliar a estrutura e saída dos programas
- Programar problemas propostos

Nome Completo: *

Harllem Alves do Nascimento

Entrada e Saída

1 de 1 pontos

Comandos: printf e scanf

printf

Comando de saída. Sintaxe:

printf("Olá mundo novo");

O texto deve estar entre aspas. Atenção para o ponto e vírgula após cada instrução na linguagem C

scanf

Comando de entrada. Sintaxe:

scanf("%d", $var1);

Antes observe que a variável em c obedece a seguinte expressão letra(letra|dígito)* Ou seja, letra seguida de letra ou número. No exemplo nossa variável é var1

No scanf observe o especificador de formato %d. Eles são usados para formatar as entradas e saídas:

%d - inteiro
%f - real
%c - caractere
%s - string de caracteres

Além do especificador de formato, temos o & que indica que a variável será referenciada por seu endereço, e não pelo seu valor.

Entrada e Saída

Exemplo:

```
#include <stdio.h>

main(){

int a;
int b;

printf("Digite o primeiro valor: \n ");
scanf("%d", &a);
printf("Digite o segundo valor: \n ");
scanf("%d", &b);

int soma = a+b;

printf("A soma de %d + %d é %d:", b, a, soma);

return 0;
}
```

Observe no trecho as seguintes linhas:

- #include <stdio.h> diretiva para utilizar comandos e funções nativas das bibliotecas. Essa se refere a entrada e saída
- int soma = a+b; no processamento e saída das variáveis não precisa utilizar o &, aqui trabalha-se com o valor dessas variáveis
- printf("A soma de %d + %d é %d:", b, a, soma); A saída deve conter os especificadores de formato também, como podem ver.

✓ Considerando o último código, qual será a saída correta para as entradas 6 e 3? * 1/1

- ( ) A soma de 6 + 3 é 9:
- (•) A soma de 3 + 6 é 9: ✓
- ( ) o código não irá compilar

Feedback

Exibimos b antes de a pra verificar se estava atento!

String 0 de 1 pontos

Algumas considerações sobre cadeias de caracteres

Para entradas de texto podemos usar o especificador %s e a função gets();

Exemplo:

```
#include <stdio.h>

main(){

char nome[10];

printf("Digite o seu nome: \n ");
gets(nome);

printf("Olá %s, ", nome);

return 0;
}
```

Linhas:
- char nome[10]; declaração da cadeia de caracteres. Deve-se informar o tamanho; aqui nosso nome possui até 10 caracteres
- gets(nome); função para leitura/entrada do teclado
- printf("Olá %s, ", nome); saída no formato string

Observe cada variável

```
char S1[] = "";
char S2[] = "Rio de Janeiro";
char S3[81];
char S4[81] = "Rio";
```

- **S1** cadeia de caracteres vazia (armazena o caractere '\0')
- **S2** armazena cadeia de 14 caracteres (em vetor com 15 elementos)
- **S3** armazena cadeia com até 80 caracteres dimensionada com 81 elementos, mas não inicializada
- **S4** armazena cadeias com até 80 caracteres com os primeiros quatro elementos atribuídos na declaração {'R', 'i', 'o', '\0'};

Lendo com scanf

É possível utilizar o **scanf** para ler caracteres e cadeias de caracteres.

- O formato **%c** permite a leitura de um único caractere:

```
char a;
...
scanf("%c", &a);
...
```

- O formato **%s** permite a leitura de uma cadeia de caracteres não brancos:

```
char cidade[81];
...
scanf("%s", cidade);
...
```

Algumas funções da diretiva <string.h>

- **strlen**: determina o comprimento de uma cadeia;
- **strcpy**: copia uma cadeia origem para outra destino;
- **strcat**: concatena duas cadeias;
- **strcmp**: compara duas cadeias;

A função strcmp retorna um valor inteiro ao comparar duas strings. Caso o retorno seja 0 as duas cadeias serão idênticas. Complete o código a seguir, como solicitado na questão, para verificar se a entrada do programa é igual a senha definida.

```
#include <stdio.h>
#include <string.h>

main( )
{

char senha[] = "password";
char tentativa[10];
int retorno;

printf("Digite a entrada \n");
gets(tentativa);

retorno = strcmp(senha,tentativa);

//insira o seu código

return(0);
}
```

✗ Observando o código anterior, verifique o retorno da função e caso seja 0 exiba a mensagem: "Senha correta". Se o retorno não for 0 seu programa não deverá apresentar nenhuma mensagem. Use o comando if para verificar e exibir a mensagem. Obs.: insira como resposta apenas o comando if com a verificação e a mensagem a ser exibida. * ...0/1

```
if ( _expressão_booleana_ )
{
   /* Bloco de comandos */
}
```

if(retorno == 0){printf("Senha Correta!\n\n");}else{printf("Senha Incorreta!\n\n");} ✗

Respostas corretas

if (retorno == 0){ printf("Senha correta");}

if (retorno == 0){ printf("senha correta");

if (retorno == 0){ printf("Senha Correta");}

if (retorno == 0) { printf("Senha correta"); }

if(retorno == 0){printf("Senha correta");}

if(retorno==0){printf("Senha correta");}

if (retorno == 0){ printf("Senha correta"); }

if (retorno == 0) { printf("Senha correta"); }

Estruturas de Seleção

2 de 2 pontos

if...else... switch () case:

if...else

A sintaxe do if else dentro do programa C segue a seguinte estrutura:

```
if (<condição>){
<instruções>
}else{
<instruções>
}
```

Essa estrutura também pode ser aninhada da seguinte forma:

```
if (<condição>){
<instruções>
}else if(<condição>){
<instruções>
}else if(<condição>){
<instruções>
}else{
<instruções>
}
```

✓ Analise o código a seguir e marque a alternativa que possui a saída para o programa, considerando as seguintes entradas respectivamente: 8 e 8 1/1

*

```
#include<stdio.h>
int main() {
    int p1,p2;
    float media;
    printf("Prova de Frec 1=");
    scanf("%d",&p1);
    printf("Prova de Frec 2=");
    scanf("%d",&p2);
    media = ( p1 + p2 ) / 2;
    if (media >= 14)
        printf("Aprovado");
    else if (media < 7)
        printf("Reprovado");
     else
        printf("Prova Final");
 }
```

- ○ Aprovado
- ○ Reprovado
- ◉ Prova Final ✓
- ○ Nenhuma das alternativas

switch

Quando se tem um número de opções, por exemplo, podemos usar essa esturtura. A diferença é que diferente do if não podemos usar as operações lógicas. Aqui precisamos do valor!

Sintaxe:

switch (<operador>){
case '<valor>': <instrução>;
break;
case '<próximovalor>': <instrução>;
break;
default: <instrução>;
}

A função do break é sair da estrutura de seleção assim que o operador seja localizado em um dos casos. Default é utilizado para quando nenhum dos casos é satisfeito, ou seja, nenhuma das opções (valores) foram um dos esperados nos casos.

Observe a verificação da variável char op.

```
#include <stdio.h>
main ( )
{
    char op;
    float num 1, num 2;

      printf (" digite um n.o, um operador e um n.o");
      scanf (" %f  %c %f",  &num1,  &op,  &num2);
      switch (op) {
          case '+':
              printf (" = %f", num 1 + num 2);
              break:
          case  '-':
              printf (" = %f", num 1 - num 2);
              break;
          default:
              printf (" operador inválido");
      }
}
```

✓ Considerando a entrada: 5, *, 10 para o código anterior. Qual será a saída do programa? * 1/1

- = 50
- (●) operador inválido ✓
- = 15
- = 5

Estruturas de Repetição 2 de 2 pontos

for - while - do...while

Sabemos o número de vezes

Quando temos ciência da quantidade de vezes que a repetição se dará, utilizamos a estrutura de repetição FOR.

for (<inicialização>;<condição>;<incremento>){
<instruções>
}

int i;
for(i=0;i<10;i++){
printf("%d", i);
}

Nosso i é o iterador do laço, ele é incrementado sempre a cada loop (i++, mesmo que i=i+1). Nosso for irá repetir 10 vezes, pois i vai de 0 até 9 (i<10).

Podemos também aninhar as estruturas de repetição. Nesse caso o for interno executará o seu número de vezes, vezes o número de vezes do for externo.

```c
#include <stdio.h>

int main() {
  int coluna, linha, n;
  printf("Digite um numero: ");
  scanf("%d", &n);

  for (linha=1; linha <= n; linha++) {
    for (coluna=1; coluna <= n; coluna++) {
      printf("* ");
    }
    printf("\n");
  }
  return(0);
}
```

✓ Considerando a entrada 4 para o código da imagem anterior, qual será a saída? Obs.: O \n permite pular uma linha. * 1/1

- ( ) um retângulo formado pelo caractere *
- (•) um quadrado formado pelo caractere * ✓
- ( ) quatro caractere * em uma linha
- ( ) quatro caractere * , um em cada linha

while - do...while

Depende da sua decisão! Usar o while ou o do...while vai depender do momento que pretende fazer a verificação do laço.
Em ambas estruturas precisamos definir o critério de saída do laço. Isso porque utilizamos essas estruturas quando não sabemos quantas vezes o laço irá repetir.

Sintaxe:

```
while (<teste>){
<instrução>
}

do{
<instrução>
}while (<teste>);
```

Note que no while o teste é feito antes de entrar no laço, já no do...while a instrução é executada antes do teste.

Nesse exemplo temos uma variável de controle, contagem. Observem que ela é incrementada (contagem++) dentro do laço while. Ou seja, o laço irá repetir 5 vezes e a cada loop contagem = contagem+1 (contagem++).

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    double val, soma;
    int contagem;

    soma = 0.0; // inicializando o valor da soma

    contagem = 1; // inicializando o contador

    while (contagem <= 5) {
        printf("\nDigite o %do. numero: ", contagem);
        scanf("%lf", &val);
        soma += val; // soma = soma + val
        contagem++; // contagem = contagem + 1
    }
    printf("\nO resultado da soma eh: %.2f", soma);
    system("pause");
    return 0;
}
```

✓ Suponto que a entrada para o código da imagem anterior seja 5, sempre que o laço for repetido, qual será a saída? * 1/1

- (●) 25.00 ✓
- ( ) 30.00
- ( ) 20.00
- ( ) 15.00

Exercício 0 de 0 pontos

Acerte o número!

Para gerar um valor aleatório em C podemos usar a finção rand() da biblioteca <stdlib.h>. Porém o valor inicial ("semente") desta função é sempre o mesmo. Pra resolver isso pode-se inicializar o gerador de número aleatório através do srand(time(NULL)), como apresentado no código a seguir. Note que a biblioteca <time.h> também é necessária.

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>// necessário p/ as funções rand() e srand()
#inclue<stdio.h>
#include<time.h>//necessário p/ função time()

int main(void)
{
  int i;

  printf("Gerando 10 valores aleatorios:\n\n");

  /* srand(time(NULL)) objetiva inicializar o gerador de números aleatórios
  com o valor da função time(NULL). Este por sua vez, é calculado
  como sendo o total de segundos passados desde 1 de janeiro de 1970
  até a data atual.
  Desta forma, a cada execução o valor da "semente" será diferente.
  */
  srand(time(NULL));

  for (i=0; i < 10; i++)
  {
    // gerando valores aleatórios na faixa de 0 a 100
    printf("%d ", rand() % 100);
  }

  getch();
  return 0;
}
```

Considerando as funções do código anterior, rand() e srand(), crie um programa em C que gere um valor aleatório que deverá ser comparado com uma entrada do teclado. Enquanto o valor da entrada não for igual ao valor gerado aleatoriamente ou igual a zero o programa continuará solicitando a entrada. Por fim deve exibir a mensagem 'Acertou' caso a entrada seja igual ao valor aleatório ou 'Saiu', se a entrada for zero. Obs.: Poderá precisar usar os operadores lógicos: && (and), || (ou), != (diferente), == (igual). *

```
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <ctype.h>
#include <stdbool.h>
#include <string.h>

int main(void){
   system("cls");

   int valChk, randN;
   bool try, win;
   char opt[1];

   try = true;
   win = false;
   srand(time(NULL));
   randN = rand() % 10;
   setlocale(LC_ALL,"Portuguese");

   while(try){
      system("cls");
      printf("=+=+=+=+=+=+=+=+Descubra o numero!(0 a 10)=+=+=+=+=+=+=+=+\n\n");
      printf("\nDigite um numero para verificar: ");
      scanf("%d", &valChk);

      if(randN == valChk){
         printf("\nVocê Acertou!");
         win = true;
         try = false;
      }

      if(!(win)){
         printf("Tentar novamente(N para sair)?");
         scanf("%s",&opt);
         opt[0] = toupper(opt[0]);
         //Compara string com a opção
         if(strcmp(opt,"N") == 0){
            try = false;
         }
```

```
            //Limpa string
            strcpy(opt,"");
        }
    }
    printf("\n");
    return 0;
}
```