INTRODUÇÃO À PROGRAMAÇÃO | 21173

Data e hora de realização

26 de janeiro de 2023

Duração da prova

90m + 60m

Instruções

- O estudante deverá responder à prova na folha de resolução.
- A cotação é indicada junto de cada pergunta.
- A prova é individual, mas pode ser realizada com consulta. Todos os elementos consultados devem ser referenciados na prova.
- A interpretação do enunciado das perguntas também faz parte da sua resolução, pelo que, se existir alguma ambiguidade, deve indicar claramente como foi resolvida.
- A prova é constituída por 4 grupos, estando a cotação indicada em cada grupo.
- Ao resolver os grupos III e IV, pode e deve utilizar as funções definidas nos grupos anteriores, mesmo que não os tenha realizado.
- Os programas devem ser escritos em **linguagem C** podendo utilizar funções da biblioteca standard. Em anexo está uma lista com as funções da biblioteca standard mais utilizadas, não sendo necessário utilizar a primitiva #include.

Enunciado

Grupo I (3 valores)

Este grupo é constituído por 5 perguntas de escolha múltipla e um conjunto de 5 perguntas de correspondência. As perguntas devem ser respondidas no contexto da linguagem de programação C. Cada pergunta vale 0.3 valores, não existindo penalização por falha.

1. Expressão lógica A menor que B:

a)	b)	c)	d)
A-B	A*B	A<B	A<=B

2. String de formatação para mostrar um real em numeração científica:

a)	b)	c)	d)
%g	String	for	%f

3. String de formatação para mostrar um carácter:

a)	b)	c)	d)
%c	%d	char	Ciclo

4. Expressão lógica não A:

a)	b)	c)	d)
A<B	!A	A&&B	A!=B

5. Expressão numérica A a dividir por B:

a)	b)	c)	d)
A*B	A%B	A<=B	A/B

Perguntas de Correspondência:

6. É uma instrução que deve ser executada dependente de uma expressão lógica

7. É uma função que é chamada pelas ações que faz, e não por calcular uma grandeza

8. <inicialização>;
   ?ciclo?
   {
   <instrução>;
   <atualização>;
   } ?ciclo?(<condição>);

9. É uma instrução que deve ser executada até que se deixe de verificar uma condição

10. É uma variável que serve para controlar a paragem de um ciclo

Respostas possíveis:

a)	b)	c)	d)	e)
Condicional	Ciclo	do-while	Procedimento	Variável iteradora

Grupo II (3 valores)

Complete o programa em baixo, faltando definir a função *JPLanceValido*, utilizada na função main.

```
int Primo(int n)
{
    int divisor = 2;
    while (divisor * divisor <= n) {
        if (n % divisor == 0)
            return 0;
        divisor++;
    }
    return 1;
}



int main()
{
    int k, w;

    printf("\nIndique Lance (dois numeros): ");
    scanf("%d %d", &k, &w);

    // verificar a validade do lance
    if (JPLanceValido(k, w))
        printf("Lance valido");
    else
        printf("Lance invalido");
}
```

Este grupo e os dois seguintes, são baseados no jogo dos fatores primos. Este jogo necessita de um inteiro positivo K. Dois jogadores jogam à vez, de forma alternada. Em cada jogada podem **dividir ou subtrair** o número por um dos seus **fatores primos**. O resultado substitui o valor de K e passam a vez de jogar. Ganha o jogador que fizer uma jogada convertendo o número K para 1 ou 0, ou se o adversário tiver feito uma jogada inválida. O número 1 não é aqui considerado um fator primo.

Pretende-se que verifique se uma jogada de K para W é válida, estando disponível para utilização uma função que retorna se um dado número é primo. Um fator primo de K é um número que é divisível por K (resto da divisão inteira é nulo) e é primo.

Casos de execução:

Entrada	Saída
45 15	Lance valido
45 40	Lance valido
5 10	Lance invalido
10 5	Lance valido
12 4	Lance valido
12 3	Lance invalido
12 8	Lance invalido

Nos dois primeiros casos, o número 45 é dividido por 3, que é um fator primo de 45, pelo que é válido, enquanto que no segundo caso o número é subtraído por 5, também o fator primo sendo válido. O terceiro caso tem uma situação em que K é menor que W, pelo que é sempre inválido. No penúltimo caso, 12 é dividido por 4, que é divisor de 12 mas não é primo, enquanto que no último caso 12 é subtraído por 4, também divisor de 12 mas não primo, pelo que ambos os lances são inválidos.

Grupo III (3 valores)

Neste grupo pretende-se que calcule todas as jogadas possíveis, não mostrando eventuais jogadas duplicadas. Complete o código em baixo em que falta definir a função *JPJogadas*:

```
#define MAX_JOGADAS 20

int Primo(int n)
{
    int divisor = 2;
    while (divisor * divisor <= n) {
        if (n % divisor == 0)
            return 0;
        divisor++;
    }
    return 1;
}


int main()
{
    int k, n, i;
    int jogadas[MAX_JOGADAS];

    printf("\nIndique Posicao (um numero): ");
    scanf("%d", &k);

    // calcular as jogadas
    n = JPJogadas(k, jogadas);

    // mostrar as jogadas
    printf("\nJogadas (%d): ", n);
    for (i = 0; i < n; i++)
        printf("%d ", jogadas[i]);
}
```

A função deverá receber a posição atual (um valor inteiro), e um vetor de inteiros com tamanho suficiente para colocar todas as jogadas, e retornar quantas jogadas foram colocadas no vetor.

Casos de execução:

Entrada	Saída
45	`Jogadas (4): 15 42 9 40`
40	`Jogadas (4): 20 38 8 35`
5	`Jogadas (2): 1 0`
10	`Jogadas (3): 5 8 2`
12	`Jogadas (4): 6 10 4 9`
4	`Jogadas (1): 2`
8	`Jogadas (2): 4 6`

A ordem das jogadas não interessa, mas não devem ser mostradas jogadas duplicadas. Apenas os fatores primos devem ser utilizados, pelo que os números têm de ser divisíveis por K e primos, tal como no grupo II. Pode-se ver no caso 45 que há 4 jogadas, resultante dos dois fatores primos, 3 e 5, existindo duas operações por fator primo, que resultam em 4 números distintos. Para o número 40 os fatores primos são desta vez o 2 e 5, sendo idêntico ao caso anterior. O número 5 já é primo, pelo que divide ou subtrai por ele próprio, resultando em 1 e 0. Notar o caso do número 10 há 3 jogadas, dado que dividir 10 por 2 ou subtrair a 10 o número 5, ambas jogadas válidas, resultam no mesmo número. No caso do número 4 com um só fator primo, tanto a divisão como a subtração resultam no mesmo valor. O número 8 tem o mesmo fator primo, mas a divisão e subtração resultam em duas jogadas possíveis.

Grupo IV (3 valores)

Neste grupo pretende-se que faça um jogo entre dois jogadores humanos, com a interface descrita no caso de exemplo seguinte:

```
Indique Posicao (um numero): 45

Jogadas (4): 15 42 9 40
Lance 1: 15

Jogadas (4): 5 12 3 10
Lance 2: 12

Jogadas (4): 6 10 4 9
Lance 3: 13

Ganha jogador par.
```

Como se pode ver na execução, deve ser solicitado o número inicial, devendo ser mostradas as jogadas possíveis. Após essa situação deve-se solicitar o lance com o número da jogada, e no caso de ser válido avançar para a próxima jogada, caso contrário identificar o jogador que ganha. O jogador par é o jogador que joga nos lances pares, e o jogador impar é o jogador que joga nos lances impares.

Anexo - Funções standard mais utilizadas

Exemplos de chamadas:

- `printf("texto %d %g %s %c", varInt, varDouble, varStr, varChar);`
  Imprime no ecran uma string formatada, em que é substituído o **%d** pela variável inteira seguinte na lista, o **%g** pela variável real na lista, o **%s** pela variável string na lista, o **%c** pela variável caracter na lista.
- `scanf("%d", &varInt); gets(str);`
  scanf é a função inversa do **printf**, lê um inteiro e coloca o seu resultado em **varInt**, cujo endereço é fornecido. A função **gets** lê uma string para **str**.

Protótipos:

- `int atoi(char *str); float atof(char *str);`
  Converte uma string num número inteiro/real respectivamente
- `int strlen(char *str);`
  Retorna o número de caracteres da string **str**
- `strcpy(char *dest, char *str); [strcat]`
  Copia **str** para **dest**, ou junta **str** no final de **dest**, respectivamente
- `char *strstr(char *str, char *find); char *strchr(char *str, char find);`
  Retorna a primeira ocorrência de **find** em **str**, ou NULL se não existe. Na versão **strchr find** é um caracter.
- `char *strtok(char *string, char *sep); char *strtok(NULL, char *sep);`
  Retorna um apontador para uma token, delimitada por **sep**. A segunda chamada retorna a token seguinte, na mesma string, podendo-se continuar a chamar a função até que retorne NULL, o que significa que a string inicial não tem mais tokens para serem processadas.
- `sprintf(char *str, …); sscanf(char *str,…);`
  Estas funções têm o mesmo funcionamento de **printf**/**scanf**, mas os dados são colocados (ou lidos) em **str**.
- `int strcmp(char *str1, char *str2);`
  Retorna 0 se **str1** é igual a **str2**, retornando um valor negativo/positivo se uma string é maior/menor que a outra
- `int isalpha(int c); [isdigit,isalnum,islower,isupper,isprint]`
  Retorna true se **c** é uma letra / dígito numérico / letra ou dígito / minúscula / maiúscula / imprimivel.
- `void *malloc(size_t); free(void *pt);`
  malloc retorna um apontador para um bloco de memória de determinada dimensão, ou NULL se não há memória suficiente, e a função **free** liberta o espaço de memória apontado por **pt** e alocado por **malloc**
- `FILE *fopen(char *fich, char *mode); fclose(FILE *f);`
  fopen abre o ficheiro com nome **fich**, no modo **mode** ("rt" – leitura em modo texto, "wt" – escrita em modo texto), e **fclose** fecha um ficheiro aberto por **fopen**
- `fprintf(f,…); fscanf(f,…); fgets(char *str, int maxstr, FILE *f);`
  idênticos ao **printf**/**scanf** mas direccionados para o ficheiro, e **fgets** é uma versão do **gets** mas com limite máximo da string indicado em **maxstr**.
- `int feof(FILE *f);`
  feof retorna true se o ficheiro **f** está no fim, e false c.c.
- `fseek(f,posicao,SEEK_SET);`
  `fwrite/fread(registo,sizeof(estrutura),1,f);`
  funções de leitura binária (abrir em modo "rb" e "wb"). **fseek** posiciona o ficheiro numa dada posição, **fwrite**/**fread** escrevem/lêm um bloco do tipo estrutura para o endereço de memória registo.
- `int rand(); srand(int seed);`
  rand retorna um número pseudo-aleatório e **srand** inicializar a sequência pseudo-aleatória
- `time_t time(NULL); clock_t clock();`
  time retorna um número segundos que passaram desde uma determinada data, e **clock** o número de instantes (há **CLOCKS_PER_SEC** instantes por segundo)
- `double sin(double x); [cos,log,log10,sqrt] double pow(double x,double y);`
  Funções matemáticas mais usuais, com argumentos e valores retornados a double