Estruturas de Controle de Fluxo

3. Laços de repetição

Exercício 5:

Em uma determinada disciplina foram aplicadas duas avaliações com notas variando de zero a dez, e a nota final de cada aluno foi calculada com base na média aritmética entre as notas obtidas pelo mesmo. Codifique um programa, utilizando a linguagem C, que receba como entrada a quantidade de alunos existentes em uma determinada turma da disciplina em questão e as notas dos alunos. O programa deve exibir na saída padrão a maior nota final obtida na referida turma.

UNIVASF

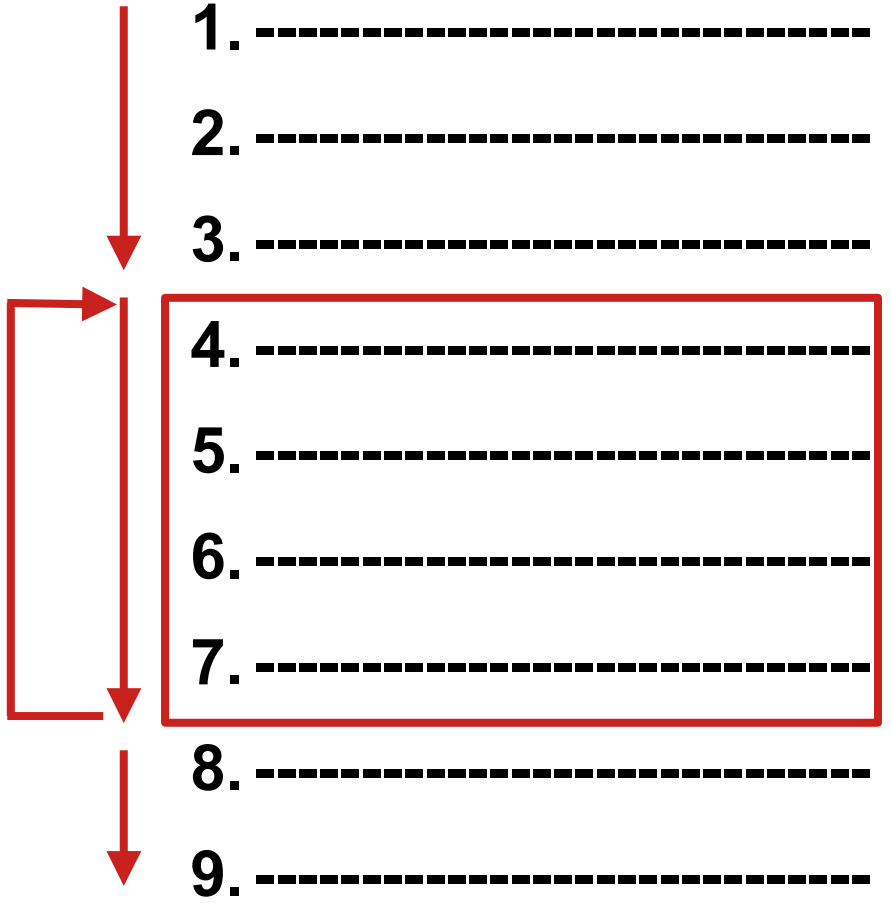
```c
#include <stdio.h>
int main() {
   float n1, n2, media, maiorMedia;
   int numeroDeAlunos, contador=1;
   printf ("Digite o numero de alunos na turma: ");
   scanf ("%d", &numeroDeAlunos);
   while (numeroDeAlunos) {
      printf ("Forneca a 1ª nota do %dº aluno entre 0 e 10: ", contador);
      scanf ("%f",&n1);
      printf ("Forneca a 2ª nota do %dº aluno entre 0 e 10:  ", contador++);
      scanf ("%f",&n2);
      if (contador==2)
         maiorMedia = (n1 + n2)/2;
      else {
         media = (n1 + n2)/2;
         if (maiorMedia < media)
            maiorMedia = media;
      }
      numeroDeAlunos--;
   }
   printf("A maior nota final obtida foi %.2f",maiorMedia);
}
```

```c
#include <stdio.h>
int main() {
   float n1, n2, media, maiorMedia;
   int numeroDeAlunos, contador=1;
   printf ("Digite o numero de alunos na turma: ");
   scanf ("%d", &numeroDeAlunos);
   while (contador<=numeroDeAlunos) {
      printf ("Forneca a 1ª nota do %dº aluno entre 0 e 10: ", contador);
      scanf ("%f",&n1);
      printf ("Forneca a 2ª nota do %dº aluno entre 0 e 10:  ", contador++);
      scanf ("%f",&n2);
      if (contador==2)
         maiorMedia = (n1 + n2)/2;
      else {
         media = (n1 + n2)/2;
         if (maiorMedia < media)
            maiorMedia = media;
      }
   }
   printf("A maior nota final obtida foi %.2f",maiorMedia);
}
```

Estruturas de Controle de Fluxo
Laços de Repetição (do while)

UNIVASF

Estruturas de Controle de Fluxo



Estruturas de Controle de Fluxo

3. Laços de repetição

```
do
 {
        <instrução1>
        . . .
        <instruçãon>
 } while (<expressão_lógica>);
```

/*faça enquanto o resultado da avaliação da expressão lógica for verdadeiro*/

UNIVASF

3. Laços de repetição

Retomando a análise do exemplo trabalhado anteriormente com o laço de repetição *while.* Porém, agora utilizando-se do laço de repetição *do while*:

```c
#include <stdio.h>
int main()
{
   float n1, n2, media, maiorMedia;
   int contador=1;
   do {
      printf ("Digite a 1ª nota do %dº aluno entre 0 e 10: ", contador);
      scanf ("%f",&n1);
      printf ("Digite a 2ª nota do %dº aluno entre 0 e 10: ", contador);
      scanf ("%f",&n2);
      if (contador==1)
         maiorMedia = (n1 + n2)/2;
      else {
         media = (n1 + n2)/2;
         if (maiorMedia < media)
            maiorMedia = media;
      }
      contador++;
   } while (contador<=4);
   printf("A maior nota final obtida foi %.2f",maiorMedia);
}
```

```c
} while (++contador<5);
```

Estruturas de Controle de Fluxo

3. Laços de repetição

```
while (<expressão_lógica>)
{
   <instrução1>
   . . .
   <instruçãon>
}
```

```
do
 {
        <instrução1>
        . . .
        <instruçãon>
 }while (<expressão_lógica>);
```

UNIVASF

Estruturas de Controle de Fluxo

3. Laços de repetição

Um aluno atento já deve ter se perguntado sobre a possibilidade de mesmo sendo solicitado um valor ao usuário, explicitando o intervalo ao qual o mesmo deve pertencer, o usuário venha a fornecer um valor fora do intervalo especificado.

Os laços de repetição podem nos ajudar nesta questão. Por exemplo, no trecho:

```
...
printf ("Digite a 1ª nota do 1º aluno entre 0 e 10: ");
scanf ("%f",&n1);
...
```

UNIVASF

Estruturas de Controle de Fluxo

Com a utilização de laços de repetição podemos reescrever o trecho anterior da seguinte forma

```
...
do
{
    printf ("Digite a 1ª nota do 1º aluno entre 0 e 10: ");
    scanf ("%f",&n1);
} while (n1>10 || n1<0);
...
```

Estruturas de Controle de Fluxo

3. Laços de repetição

Exercício 6:

Faça um programa que receba números naturais fornecidos pelo usuário, quando o usuário quiser parar a execução do programa, o mesmo fornecerá um número negativo. O programa deve retornar ao final a média dos números naturais fornecidos pelo usuário (fazer dois programas utilizando em cada um uma das estruturas de repetição vistas).

UNIVASF

```c
#include <stdio.h>
int main() {
   int num,contador=0, acumulador=0;
   do {
      printf ("Entre com um numero natural (entre com um numero negativo para sair):");
      scanf ("%d",&num);
      if (num>=0) {
         acumulador += num;
         contador++;
      }
   } while (num>=0);
   printf ("A media dos %d numeros naturais pelo usuario eh %f", contador,
   acumulador/(float)contador);
}
```

Se o primeiro valor fornecido for negativo?

```c
#include <stdio.h>
int main() {
   int num,contador=0, acumulador=0;
   do {
      printf ("Entre com um numero natural (entre com um numero negativo para sair):");
      scanf ("%d",&num);
      if (num>=0) {
         acumulador += num;
         contador++;
      }
   } while (num>=0);
   if (contador)
      printf ("A media dos %d numeros naturais pelo usuario eh %f", contador, acumulador/(float)contador);
   else
      printf ("Nao foi fornecido nenhum numero natural.");
}
```

```c
#include <stdio.h>
int main() {
   int num,contador=-1,acumulador;
   num=acumulador=0;
   while (num>=0) {
      contador++;
      printf ("Entre com um numero natural (entre comum numero negativo para sair):");
      scanf ("%d",&num);
      if (num>=0)
         acumulador += num;
   }
   if (contador)
      printf ("A media dos %d numeros naturais pelo usuario eh %f", contador,
      acumulador/(float)contador);
   else
      printf ("Nao foi fornecido nenhum numero natural");
}
```

Estruturas de Controle de Fluxo

3. Laços de repetição (continuação)

Exercício 7:

Faça um programa que receba um número natural fornecido pelo usuário e retorne seus divisores em ordem decrescente na saída padrão. (fazer dois programas distintos utilizando em cada um uma das estruturas de repetição vistas).

UNIVASF

```c
#include <stdio.h>
int main()
{
    int num, div;
    printf ("Digite um numero natural:");
    scanf ("%d", &num);
    div = num;
    do
    {
        if (num%div==0)
            printf ("%d eh divisor de %d\n", div, num);
        div--;
    }while (div);
}
```

Se o usuário fornecer um valor negativo?

```c
#include <stdio.h>
int main()
{
    int num, div;
    do {
        printf ("Digite um numero natural:");
        scanf ("%d", &num);
    } while (num<0);
    div = num;  div = num?num:1;
    do
    {
       if (num%div==0)
          printf ("%d eh divisor de %d\n", div, num);
       div--;
    }while (div);
}
```

Se num for igual a zero?

```c
#include <stdio.h>
int main() {
    int num=-1, div;
    while (num<0) {
        printf ("Digite um numero natural:");
        scanf ("%d", &num);
    }
    div = num?num:1;
    while (div)
    {
       if (num%div==0)
          printf ("%d eh divisor de %d\n", div, num);
       div--;
    }
}
```

```c
#include <stdio.h>
int main() {
    int num, div;
    do {
        printf ("Digite um numero natural:");
        scanf ("%d", &num);
    } while (num<0);
    div = num?num:1;
    while (div)
    {
        if (num%div==0)
            printf ("%d eh divisor de %d\n", div, num);
        div--;
    }
}
```

Estruturas de Controle de Fluxo
Comandos break e continue

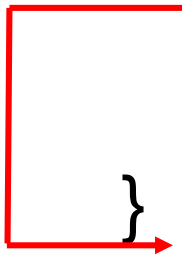
UNIVASF

Estruturas de Controle de Fluxo

4. Comando *break*

Exemplo:

```
while (x>100)
{
        x-=b*3;
        if (x<100)
            break;
        x-=y*3;
}
```



Estruturas de Controle de Fluxo

5. Comando *continue*

Exemplo:

```
while (x>100)
{
        x-=b*3;
        if (x<y)
            continue;
        x-=y*3;
}
```

Estruturas de Controle de Fluxo

Visando fomentar uma maior participação de discentes em competições acadêmicas, mais especificamente na Maratona de Programação e, também, buscando viabilizar uma possível correção parcialmente automatizada de uma avaliação da disciplina, trabalharemos com uma forma alternativa para entrada de dados para programas.

Como vimos até o momento os programas ao se utilizarem da função de entrada formatada *scanf*() efetuam leitura de dados através do teclado.

Também vimos que tudo que é digitado é armazenado em um arquivo denominado *stdin* e posteriormente lido pelas chamadas à função *scanf*().

UNIVASF

Estruturas de Controle de Fluxo

Para uma melhor compreensão, vamos analisar o programa a seguir:

```
#include <stdio.h>
int main ()
{
   int valorUm, valorDois;
   printf("Digite um valor inteiro: ");
   scanf("%d", &valorUm);
   printf("Digite outro valor inteiro: ");
   scanf("%d", &valorDois);
   printf("%d+%d=%d\n", valorUm, valorDois,
   valorUm+valorDois);
}
```

UNIVASF

Estruturas de Controle de Fluxo

Agora abra um editor de texto e digite no mesmo o conteúdo abaixo:

6 13

Após, salve o texto em um arquivo com o nome entrada.txt na pasta em que você salvou o arquivo fonte.

No prompt de comando (no Windows) ou no terminal (no Linux), na pasta em que você compilou o arquivo fonte digite:

./nomeDoArquivoBinario < entrada.txt

Estruturas de Controle de Fluxo

Será obtida a seguinte saída no monitor:

Digite um valor inteiro: Digite outro valor inteiro: 6+13=19

Isto demonstra que podemos através do símbolo < informar um arquivo que será considerado como *stdin.*

Vamos agora editar o código fonte retirando os trechos de solicitação de entradas por parte do usuário, sublinhados em vermelho no próximo slide.

UNIVASF

Estruturas de Controle de Fluxo

```c
#include <stdio.h>
int main ()
{
   int valorUm, valorDois;
   printf("Digite um valor inteiro: ");
   scanf("%d", &valorUm);
   printf("Digite outro valor inteiro: ");
   scanf("%d", &valorDois);
   printf("%d+%d=%d\n", valorUm, valorDois,
   valorUm+valorDois);
}
```

Após uma nova compilação e execução com:

./nomeDoArquivoBinario < entrada.txt

Temos a seguinte saída:

6+13=19

Estruturas de Controle de Fluxo

Com base no que foi visto, resolva o exercício a seguir:

Construa um programa que receba uma sequência de números inteiros, positivos de três dígitos, e gere como saída, para cada valor pertencente à sequência, outro número formado pelos dígitos invertidos do número lido. O final da sequencia é identificado pelo valor zero.

Exemplo de entrada:	Exemplo de saída:
123	321
901	109
530	035
0	

Dica: Observe os resultados das funções Quociente e Resto de um número por 10.

```c
#include <stdio.h>
int main()
{
   int N;
   do
   {
      scanf("%d", &N);
      if (N!=0)
         printf ("%03d\n", (N%10*100) + (N/10%10*10) + (N/10/10));
   }while(N);
}
```

```c
#include <stdio.h>
int main()
{
   int N;
   do
   {
      scanf("%d", &N);
      if (!N )
         break;
      printf ("%03d\n", (N%10*100) + (N/10%10*10) + (N/10/10));
   }while(1);
}
```

Estruturas de Controle de Fluxo

Exercício 8:

Com base no que foi visto, resolva o problema intitulado “Jogo de Varetas” aplicado na primeira fase da Maratona de Programação no ano de 2007.

Há muitos jogos divertidos que usam pequenas varetas coloridas. A variante usada neste problema envolve a construção de retângulos. O jogo consiste em, dado um conjunto de varetas de comprimentos variados, desenhar retângulos no chão, utilizando as varetas como lados dos retângulos, sendo que cada vareta pode ser utilizada em apenas um retângulo, e cada lado de um retângulo é formado por uma única vareta. Nesse jogo, duas crianças recebem dois conjuntos iguais de varetas. Ganha o jogo a criança que desenhar o maior número de retângulos com o conjunto de varetas.

UNIVASF

Dado um conjunto de varetas de comprimentos inteiros, você deve escrever um programa para determinar o maior número de retângulos que é possível desenhar.

Entrada

A entrada contém vários casos de teste. A primeira linha de teste contém um inteiro N que indica o número de diferentes comprimentos de varetas (1 ≤ N ≤ 1.000) no conjunto. Cada uma das N linhas seguintes contém dois números inteiros Ci e Vi , representando respectivamente um comprimento (1 ≤ Ci ≤ 10.000) e o número de varetas com esse comprimento (1 ≤ Vi ≤ 1.000). Cada comprimento de vareta aparece no máximo uma vez em um conjunto de teste (ou seja, os valores Ci são distintos). O final da entrada é indicado por N = 0.

Saída

Para cada caso de teste da entrada seu programa deve produzir uma única linha na saída, contendo um número inteiro, indicando o número máximo de retângulos que podem ser formados com o conjunto de varetas dado.

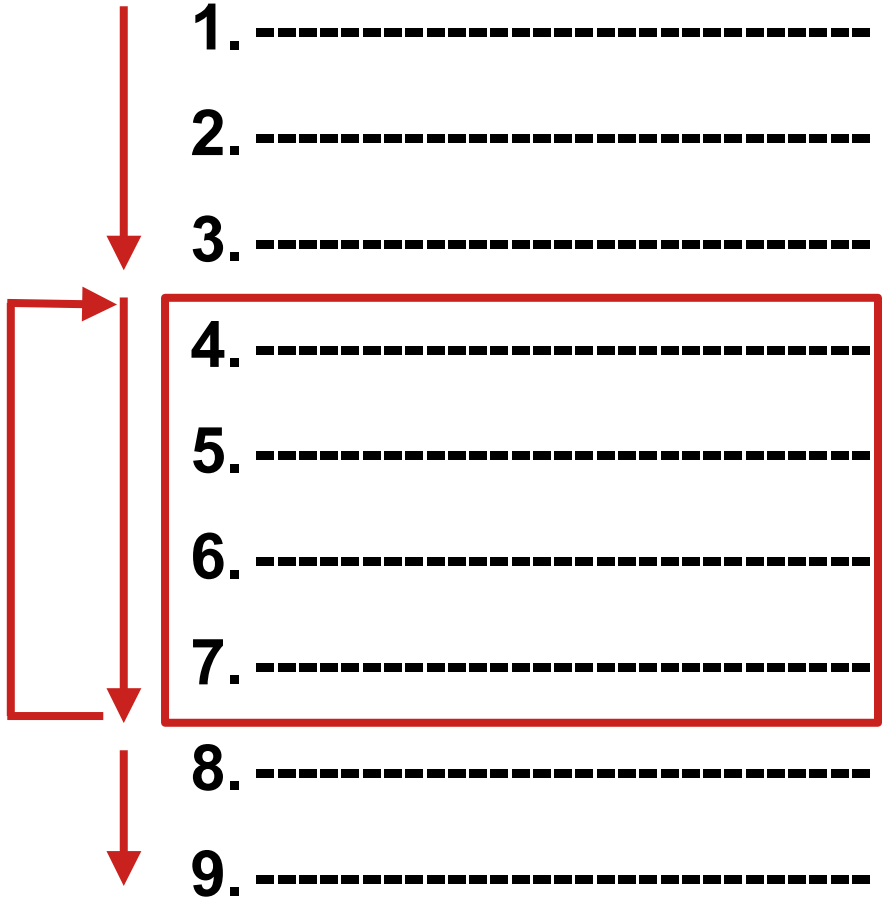
UNIVASF

Exemplo de Entrada	Exemplo de Saída
1 10 7 4 50 2 40 2 30 4 60 4 5 15 3 6 3 12 3 70 5 71 1 0	1 3 2

Estruturas de Controle de Fluxo
Laços de Repetição (for)

UNIVASF

Estruturas de Controle de Fluxo



Estruturas de Controle de Fluxo

3. Laços de repetição

```
. . .
for (<instruçãoX>;<expressão_lógica>;<instruçãoY>)
{
        <instruçãoZ>
        . . .
}
. . .
```

UNIVASF

Estruturas de Controle de Fluxo

3. Laços de repetição

```
for (i=0,j=0;i<10&&j<20;i++,j+=2)
{
        . . .
}

for (i=0;i<1000;i++);
printf ("\nEscreve qualquer coisa!\n");
/*O printf anterior não está contido no for!*/
```

Estruturas de Controle de Fluxo

3. Laços de repetição

Exemplo:

O programa a seguir recebe um número natural e retorna o seu fatorial na saída padrão.

UNIVASF

```c
#include <stdio.h>
int main()
{
   int i,num,fat=1;  long int i,num,fat=1;
   do
   {
      printf ("Digite um numero natural:");
      scanf ("%ld",&num);
   }while(num<0);
   for (i=2;i<=num;i++)  for (i=2; i<=num; fat*=i, i++);
      fat*=i;                 for (i=2; i<=num; fat*=i++);
   printf("\nO fatorial do numero%ldeh:%ld\n", num, fat);
}
```

Estruturas de Controle de Fluxo

3. Laços de repetição

Exercício 9:

Construa um programa que leia da entrada padrão uma sequência de números inteiros positivos, onde o final da sequência é identificado pelo valor zero. O programa deve determinar se os números recebidos são ou não quadrados perfeitos.

Obs.1: As entradas devem ser validadas.

Obs.2: Um número é um quadrado perfeito quando tem um número inteiro como raiz quadrada.

UNIVASF