

MC102 – Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Segundo Semestre de 2013

Roteiro

1 Simulação de código

2 Comandos de repetição

3 while (condição) { comandos }

4 do { comandos } while (condição)

5 for (início; condição; passo) { comandos }

6 Comandos break e continue

Simulação de código

- Nem sempre os resultados produzidos por um programa são os esperados.
- Isso pode ser devido a alguns motivos, entre os quais:
 - ▶ Erros de codificação (sintaxe): uma ou mais instruções escritas incorretamente.
 - ▶ Erros de lógica (semântica): erro no planejamento dos passos usados para resolver o problema (algoritmo).
- Algumas estratégias para detectar erros em programas são:
 - ▶ Simulação Automática: utilizando-se um depurador (ex.: gdb).
 - ▶ Simulação Manual: utilizando-se papel e caneta (ou printf's).

Simulação Manual

- Processo simples envolvendo apenas 2 passos:
 - ▶ “Alocação de memória”
 - ▶ “Execução passo a passo”

Simulação Manual

- Execução em memória:
 - ▶ Suponha o seguinte código:
 1. `int divisor, dividendo;` ← Último executado
 2. `float resultado;` ← Próximo Comando
 3. `divisor = 10;`
 4. `dividendo = 13;`
 5. `resultado = dividendo / divisor;`
- Após “executar” a linha 1 (alocação de memória):

Tipo	<code>int</code>	<code>int</code>
Nome	<code>divisor</code>	<code>dividendo</code>
Valor	?	?

Simulação Manual

- Execução em memória:
 - ▶ Suponha o seguinte código:
 1. `int divisor, dividendo;`
 2. `float resultado;` ← Último executado
 3. `divisor = 10;` ← Próximo Comando
 4. `dividendo = 13;`
 5. `resultado = dividendo / divisor;`
- Após “executar” a linha 2 (alocação de memória):

Tipo	int	int	float
Nome	divisor	dividendo	resultado
Valor	?	?	?

Simulação Manual

- Execução em memória:
 - ▶ Suponha o seguinte código:
 1. `int divisor, dividendo;`
 2. `float resultado;`
 3. `divisor = 10;` ← Último executado
 4. `dividendo = 13;` ← Próximo Comando
 5. `resultado = dividendo / divisor;`
- Após “executar” a linha 3:

Tipo	<code>int</code>	<code>int</code>	<code>float</code>
Nome	<code>divisor</code>	<code>dividendo</code>	<code>resultado</code>
Valor	10	?	?

Simulação Manual

- Execução em memória:
 - ▶ Suponha o seguinte código:
 1. `int divisor, dividendo;`
 2. `float resultado;`
 3. `divisor = 10;`
 4. `dividendo = 13;` ← Último executado
 5. `resultado = dividendo / divisor;` ← Próximo Comando
- Após “executar” a linha 4:

Tipo	<code>int</code>	<code>int</code>	<code>float</code>
Nome	<code>divisor</code>	<code>dividendo</code>	<code>resultado</code>
Valor	10	13	?

Simulação Manual

- Execução em memória:
 - ▶ Suponha o seguinte código:
 1. `int divisor, dividendo;`
 2. `float resultado;`
 3. `divisor = 10;`
 4. `dividendo = 13;`
 5. `resultado = dividendo / divisor;` ← Último executado
- Após “executar” a linha 5:

Tipo	<code>int</code>	<code>int</code>	<code>float</code>
Nome	<code>divisor</code>	<code>dividendo</code>	<code>resultado</code>
Valor	10	13	1.0

Simulação Manual

- Execução em memória:
 - ▶ Suponha o seguinte código:
 1. `int divisor, dividendo;`
 2. `float resultado;`
 3. `divisor = 10;`
 4. `dividendo = 13;`
 5. `resultado = dividendo / divisor;`
- Término da execução (não há mais comandos).

Tipo	<code>int</code>	<code>int</code>	<code>float</code>
Nome	<code>divisor</code>	<code>dividendo</code>	<code>resultado</code>
Valor	10	13	1.0

Simulação Manual

- Execução em memória:
 - ▶ Suponha o seguinte código (corrigido):
 1. `int divisor, dividendo;`
 2. `float resultado;`
 3. `divisor = 10;`
 4. `dividendo = 13;`
 5. `resultado = (float) dividendo / (float) divisor;`
- Término da execução (não há mais comandos).

Tipo	int	int	float
Nome	divisor	dividendo	resultado
Valor	10	13	1.3

Comandos de Repetição

- Até agora, vimos como escrever programas capazes de executar comandos de forma sequencial e, se necessário, tomar decisões com relação a executar ou não um bloco de comandos.
- Entretanto, eventualmente é necessário executar um bloco de comandos várias vezes para obter o resultado desejado.

Exemplo:

Calcule a divisão inteira de dois números usando apenas soma e subtração.

Solução

- Duas variáveis: temporário e contador
 - 1 temporário = dividendo
 - 2 contador = 0
 - 3 Enquanto (temporário \geq divisor) faça:
 - 1 temporário = temporário - divisor
 - 2 contador = contador + 1
 - 4 Exiba contador

Por quê?

Ao final da execução, a variável contador armazena o valor da divisão inteira de dividendo por divisor.

Comandos de Repetição

- Ex.: programa que imprime todos os números de 1 a 4.

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

```
int main() {
```

```
    printf("1\n");
```

```
    printf("2\n");
```

```
    printf("3\n");
```

```
    printf("4\n");
```

```
    return 0;
```

```
}
```

Comandos de Repetição

- Ex.: programa que imprime todos os números de 1 a 100.

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

```
int main() {
```

```
    printf("1\n");
```

```
    printf("2\n");
```

```
    printf("3\n");
```

```
    printf("4\n");
```

```
    ...
```

```
    printf("100\n");
```

```
    return 0;
```

```
}
```

Comandos de Repetição

- Ex.: programa que imprime todos os números de 1 a $n \leq 100$.

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    if (n >= 1)
        printf("1\n");
    if (n >= 2)
        printf("2\n");
    if (n >= 3)
        printf("3\n");
    ...
    if (n >= 100)
        printf("100\n");

    return 0;
}
```



```
while (condição) { comandos }
```

- Estrutura:

```
while (condição)
    comando;
```

- Ou:

```
while (condição) {
    comandos;
}
```

- Passo 1: Se condição for verdadeira, vai para Passo 2. Caso contrário, encerra o bloco de repetição (while).
- Passo 2: Executa comandos.
- Passo 3: Volta para o Passo 1.

Imprimir os 100 primeiros números inteiros positivos

```
#include <stdio.h>

int main() {
    int i = 1;

    while (i <= 100) {
        printf("%d\n", i++);
    }

    return 0;
}
```

Imprimir os n primeiros números inteiros positivos

```
#include <stdio.h>

int main() {
    int i = 1;
    int n;

    scanf("%d", &n);

    while (i <= n) {
        printf("%d\n", i++);
    }

    return 0;
}
```

Imprimir as n primeiras potências de 2 ($n \geq 1$)

```
#include <stdio.h>

int main() {
    int i = 1, pot = 2;
    int n;

    scanf("%d", &n);

    while (i <= n) {
        printf("2^%d = %d\n", i++, pot);
        pot *= 2;
    }

    return 0;
}
```

```
while (condição) { comandos }
```

- O que acontece se a condição for falsa na primeira vez que ela for testada?

Exemplo:

```
while (a != a) {  
    a++;  
}
```

- O que acontece se a condição for sempre verdadeira?

Exemplo:

```
while (a == a) {  
    a++;  
}
```

```
while (condição) { comandos }
```

- O que acontece se a condição for falsa na primeira vez que ela for testada?

Exemplo:

```
while (a != a) {  
    a++;  
}
```

Resposta: o programa nunca entra na repetição.

- O que acontece se a condição for sempre verdadeira?

Exemplo:

```
while (a == a) {  
    a++;  
}
```

Resposta: o programa entra na repetição e nunca sai (*loop* infinito).

```
do { comandos } while (condição)
```

- Estrutura:

```
do
  comando;
while (condição);
```

- Ou:

```
do {
  comandos;
} while (condição);
```

- Passo 1: Executa comandos.
- Passo 2: Se condição for verdadeira, volta para Passo 1.
- Diferença em relação ao `while`: sempre executa comandos na primeira iteração. Teste é feito após a execução dos comandos.

Imprimir os 100 primeiros números inteiros positivos

```
#include <stdio.h>

int main() {
    int i = 1;

    do {
        printf("%d\n", i++);
    } while (i <= 100);

    return 0;
}
```


Imprimir os n primeiros números inteiros positivos

```
#include <stdio.h>

int main() {
    int i = 1;
    int n;

    scanf("%d", &n);

    do {
        printf("%d\n", i++);
    } while (i <= n);

    return 0;
}
```

- O que acontece quando o valor fornecido for 0 ($n = 0$)?

Imprimir as n primeiras potências de 2 ($n \geq 1$)

```
#include <stdio.h>

int main() {
    int i = 1, pot = 2;
    int n;

    scanf("%d", &n);

    do {
        printf("2^%d = %d\n", i++, pot);
        pot *= 2;
    } while (i <= n);

    return 0;
}
```

- O que acontece quando o valor fornecido for 0 ($n = 0$)?

Calcular a soma de parcelas

```
#include <stdio.h>

int main() {
    int total = 0;
    int parcela;

    do {
        printf("Entre com o valor da parcela: ");
        scanf("%d", &parcela);
        total += parcela;
    } while (parcela);

    printf("Valor total = %d\n", total);

    return 0;
}
```

```
for (início; condição; passo) { comandos }
```

- Estrutura:

```
for (início; condição; passo)
    comando;
```

- Ou:

```
for (início; condição; passo) {
    comandos;
}
```

- Início: Zero ou mais atribuições, separadas por “,”.
- Condição: Executa o bloco enquanto a condição for verdadeira.
- Passo: Zero ou mais comandos, separados por “,”.

```
for (início; condição; passo) { comandos }
```

- Passo 1: Executa início.
- Passo 2: Se condição for verdadeira, vai para Passo 3. Caso contrário, encerra o bloco de repetição (for).
- Passo 3: Executa comandos.
- Passo 4: Executa passo.
- Passo 5: Volta ao Passo 2.

O for é equivalente à seguinte construção utilizando o while:

```
início;  
while (condição) {  
    comandos;  
    passo;  
}
```

Imprimir os 100 primeiros números inteiros positivos

```
#include <stdio.h>

int main() {
    int i;

    for (i = 1; i <= 100; i++)
        printf("%d\n", i);

    return 0;
}
```

Imprimir os n primeiros números inteiros positivos

```
#include <stdio.h>

int main() {
    int i;
    int n;

    scanf("%d", &n);

    for (i = 1; i <= n; i++)
        printf("%d\n", i);

    return 0;
}
```

Imprimir as n primeiras potências de 2 ($n \geq 1$)

```
#include <stdio.h>

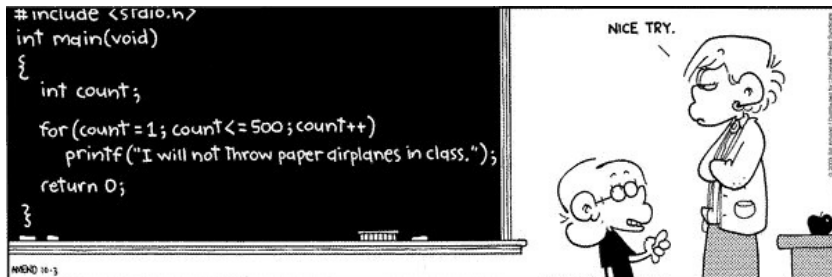
int main() {
    int i, n, pot;

    scanf("%d", &n);

    for (i = 1, pot = 2; i <= n; i++, pot *= 2)
        printf("2^%d = %d\n", i, pot);

    return 0;
}
```


I'll not throw paper airplanes in class



Comando break

O comando `break` faz com que a execução de um laço de repetição seja finalizada, passando a execução para o próximo comando após o laço.

```
int i;
for (i = 1; i <= 10; i++) {
    if (i > 4)
        break;
    printf("%d\n",i);
}
printf("Fim do programa.\n");
```

O que será impresso?

1
2
3
4

Fim do programa

Comando break

O comando `break` faz com que a execução de um laço de repetição seja finalizada, passando a execução para o próximo comando após o laço.

```
int i;
for (i = 1; i <= 10; i++) {
    if (i > 4)
        break;
    printf("%d\n",i);
}
printf("Fim do programa.\n");
```

O que será impresso?

1
2
3
4

Fim do programa

Comando continue

O comando `continue` faz com que a execução da iteração corrente do laço de repetição seja finalizada, passando a execução para a próxima iteração do laço.

```
int i;
for (i = 1; i <= 5; i++) {
    if (i == 3)
        continue;
    printf("%d\n",i);
}
printf("Fim do programa.\n");
```

O que será impresso?

1
2
4
5

Fim do programa

Comando continue

O comando `continue` faz com que a execução da iteração corrente do laço de repetição seja finalizada, passando a execução para a próxima iteração do laço.

```
int i;
for (i = 1; i <= 5; i++) {
    if (i == 3)
        continue;
    printf("%d\n",i);
}
printf("Fim do programa.\n");
```

O que será impresso?

1
2
4
5

Fim do programa

Comando continue

O comando `continue` faz com que a execução da iteração corrente do laço de repetição seja finalizada, passando a execução para a próxima iteração do laço.

```
int i;
for (i = 1; i <= 5; i++) {
    if (i == 3)
        continue;
    printf("%d\n",i);
}
printf("Fim do programa.\n");
```

O que será impresso?

1
2
4
5

Fim do programa

Exercício

- Escreva um programa que lê um número inteiro $n \geq 1$ e imprima os divisores de n .

Exercício

- Escreva um programa que imprima um menu com o nome de 4 pratos e uma quinta opção para sair do programa. O programa deve imprimir a descrição do prato solicitado e deve terminar quando a quinta opção for escolhida.

Exercício

- Escreva um programa que lê um número inteiro $n \geq 1$ e que compute e imprima o valor

$$\sum_{i=1}^n i.$$

Observação: Não use fórmulas, tal como a soma de uma progressão aritmética.

Exercício

- Escreva um programa que lê um número inteiro $n \geq 1$ e imprima os valores

$$\sum_{i=1}^j i$$

para j de 1 até n , um valor por linha.

Exercício

- Usando o Algoritmo de Euclides, calcule o MDC (Máximo Divisor Comum) de dois números inteiros positivos quaisquer.