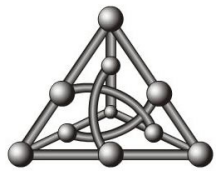
Primeiros Programas

Profa. Graziela Santos de Araújo

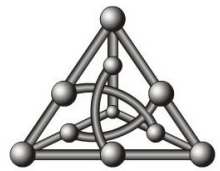
Faculdade de Computação
Universidade Federal de Mato Grosso do Sul

Algoritmos e Programação II



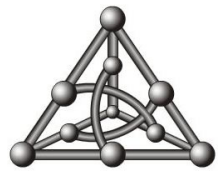
Conteúdo da aula

1. Primeiro programa
2. Próximo programa
3. Documentação
4. Entrada e saída
5. Simulação passo a passo da execução de programas
6. Exercícios



Primeiro programa

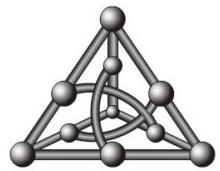
```
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```



Primeiro programa

Importante!

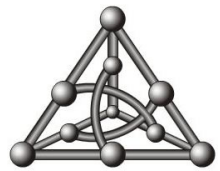
- letras minúsculas e maiúsculas são diferentes na linguagem C;
- não há distinção de onde começamos digitar nosso programa: adicionamos espaços em pontos estratégicos para facilitar a leitura do programa (**recuo** ou **indentação**);



Primeiro programa

Importante!

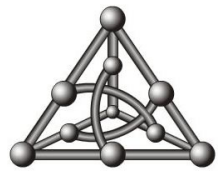
- letras minúsculas e maiúsculas são diferentes na linguagem C;
- não há distinção de onde começamos digitar nosso programa: adicionamos espaços em pontos estratégicos para facilitar a leitura do programa (**recuo** ou **indentação**);



Primeiro programa

```c
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```

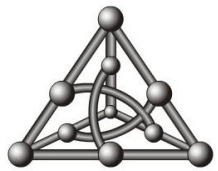
- fornece informações ao compilador sobre a função de saída de dados de nome **`printf`**
- é provavelmente incluída em todo programa que escrito na linguagem C



Primeiro programa

```c
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```

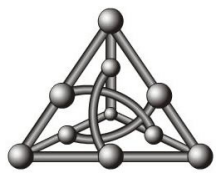
- fornece informações ao compilador sobre a função de saída de dados de nome **`printf`**
- é provavelmente incluída em todo programa que escrito na linguagem C



Primeiro programa

```c
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```

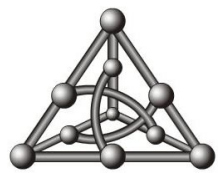
- informa ao compilador onde o programa inicia de fato
- indica que `main`, do inglês *principal*, é o início do programa principal
- o trecho iniciando nesta linha é na verdade uma função da linguagem C, que não recebe valores de entrada () ou (`void`) e devolve um valor do tipo inteiro (`int`)



Primeiro programa

```c
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```

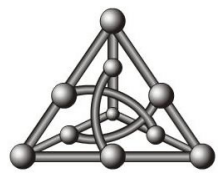
- informa ao compilador onde o programa inicia de fato
- indica que `main`, do inglês *principal*, é o início do programa principal
- o trecho iniciando nesta linha é na verdade uma função da linguagem C, que não recebe valores de entrada () ou (`void`) e devolve um valor do tipo inteiro (`int`)



Primeiro programa

```c
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```

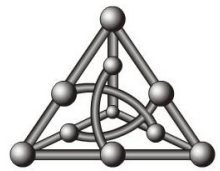
- informa ao compilador onde o programa inicia de fato
- indica que `main`, do inglês *principal*, é o início do programa principal
- o trecho iniciando nesta linha é na verdade uma função da linguagem C, que não recebe valores de entrada () ou (`void`) e devolve um valor do tipo inteiro (`int`)



Primeiro programa

```
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```

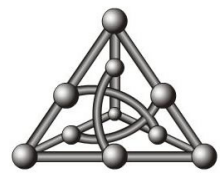
- indica o início do corpo do programa



Primeiro programa

```
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```

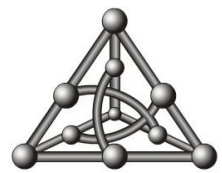
- chamada à função **`printf`**
- passamos um único **argumento** à função **`printf`**: a sequência de símbolos ou de caracteres **`"Programar é bacana!\n"`**
- todos os símbolos dessa sequência são mostrados na saída, a menos de **`"`** e **`\n`**



Primeiro programa

```
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```

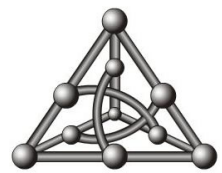
- chamada à função **`printf`**
- passamos um único **argumento** à função **`printf`**: a sequência de símbolos ou de caracteres **`"Programar é bacana!\n"`**
- todos os símbolos dessa sequência são mostrados na saída, a menos de **`"`** e **`\n`**



Primeiro programa

```
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```

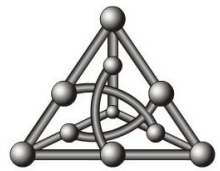
- chamada à função **`printf`**
- passamos um único **argumento** à função **`printf`**: a sequência de símbolos ou de caracteres **`"Programar é bacana!\n"`**
- todos os símbolos dessa sequência são mostrados na saída, a menos de **`"`** e **`\n`**



Primeiro programa

```
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```

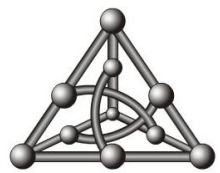
- termina a execução do programa



Primeiro programa

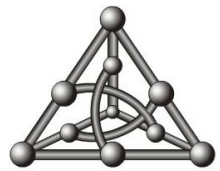
```
#include <stdio.h>

int main()
{
   printf("Programar é bacana!\n");

   return 0;
}
```

- fim do corpo do programa



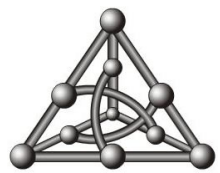
Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```



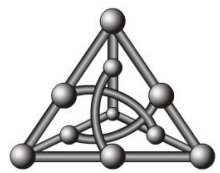
Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```



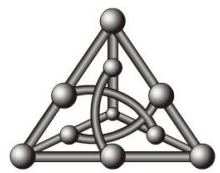
Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```



Próximo programa

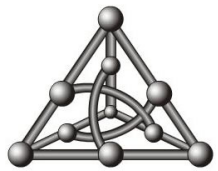
```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

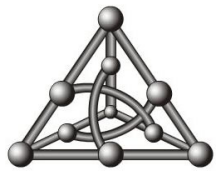
- **declaração de variáveis**: 3 variáveis do tipo inteiro são declaradas
- 3 compartimentos de memória são reservados pelo computador para armazenamento de informações (números inteiros)



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

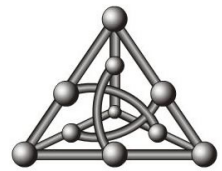
- **declaração de variáveis**: 3 variáveis do tipo inteiro são declaradas
- 3 compartimentos de memória são reservados pelo computador para armazenamento de informações (números inteiros)



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

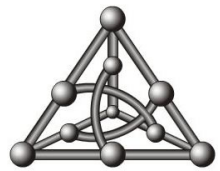
- a cada compartimento é associado um nome: **num1** , **num2** e **soma**
- esses compartimentos são conhecidos como **variáveis**: seu conteúdo pode variar durante a execução de um programa



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

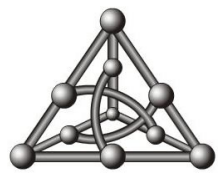
- a cada compartimento é associado um nome: **num1** , **num2** e **soma**
- esses compartimentos são conhecidos como **variáveis**: seu conteúdo pode variar durante a execução de um programa



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

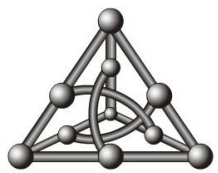
- **atribuição** do valor do tipo inteiro 25 para a variável `num1`
- **operador de atribuição** da linguagem C é `=`



Próximo programa

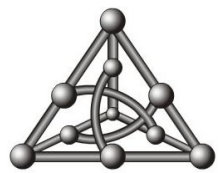
```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

- **atribuição** do valor do tipo inteiro 25 para a variável `num1`
- **operador de atribuição** da linguagem C é `=`



Próximo programa

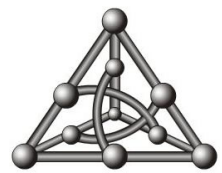
```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

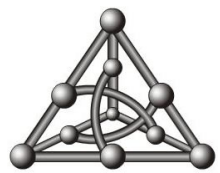
- atribuição de um valor para a variável **soma**
- atribuição do resultado da avaliação da **expressão aritmética** **num1 + num2**



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

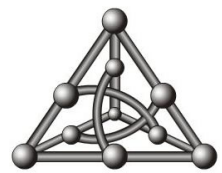
- atribuição de um valor para a variável **`soma`**
- atribuição do resultado da avaliação da **expressão aritmética** **`num1 + num2`**



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

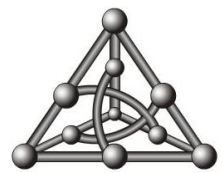
- o computador consulta o conteúdo das variáveis **num1** e **num2**
- realiza a operação de adição com os dois valores obtidos dessas variáveis e
- atribui o resultado à variável **soma**



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

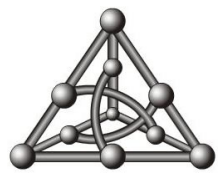
- o computador consulta o conteúdo das variáveis **num1** e **num2**
- realiza a operação de adição com os dois valores obtidos dessas variáveis e
- atribui o resultado à variável **soma**



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

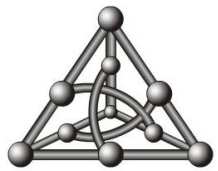
- o computador consulta o conteúdo das variáveis **num1** e **num2**
- realiza a operação de adição com os dois valores obtidos dessas variáveis e
- atribui o resultado à variável **soma**



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

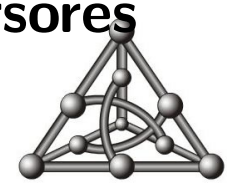
- chamada à função **`printf`** com 4 argumentos separados por vírgula



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

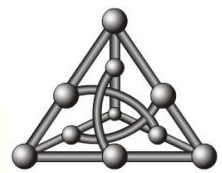
- o primeiro argumento é a **cadeia de caracteres de formatação**, contendo não apenas caracteres a serem impressos na saída, mas símbolos especiais, iniciados com `%`, conhecidos como **conversores de tipo** da linguagem C



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

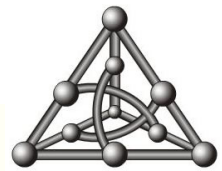
- um símbolo `%d` permite que um número inteiro seja mostrado na saída
- 3 conversores `%d`: a cada um deles está associado uma das variáveis `num1`, `num2` e `soma`, na ordem em que aparecem
- essas variáveis são os argumentos restantes da função `printf`



Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

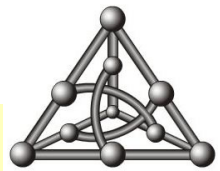
- um símbolo `%d` permite que um número inteiro seja mostrado na saída
- 3 conversores `%d`: a cada um deles está associado uma das variáveis `num1`, `num2` e `soma`, na ordem em que aparecem
- essas variáveis são os argumentos restantes da função `printf`



Próximo programa

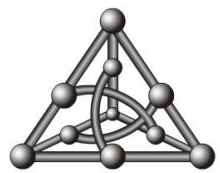
```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```

- um símbolo `%d` permite que um número inteiro seja mostrado na saída
- 3 conversores `%d`: a cada um deles está associado uma das variáveis `num1`, `num2` e `soma`, na ordem em que aparecem
- essas variáveis são os argumentos restantes da função `printf`



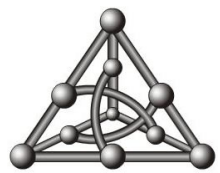
Próximo programa

```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```



Próximo programa

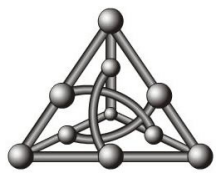
```c
#include <stdio.h>

int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```



Documentação

Segundo o professor Paulo Feofiloff, uma boa documentação de um programa significa:

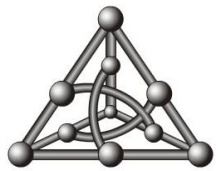
- inserir comentários apropriados no código de modo a explicar *o que* cada uma das funções que compõem o programa faz;
- a documentação de uma função é um pequeno manual que dá instruções precisas e completas sobre o uso da função;
- uma boa documentação não se preocupa em explicar *como* uma função faz o que faz, mas sim *o que* ela faz de fato, informando quais são os valores de entrada da função, quais são os valores de saída, quais as relações que esses valores que entram e saem da função e as transformações pela função realizadas.



Documentação

Segundo o professor Paulo Feofiloff, uma boa documentação de um programa significa:

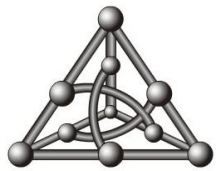
- inserir comentários apropriados no código de modo a explicar *o que* cada uma das funções que compõem o programa faz;
- a documentação de uma função é um pequeno manual que dá instruções precisas e completas sobre o uso da função;
- uma boa documentação não se preocupa em explicar *como* uma função faz o que faz, mas sim *o que* ela faz de fato, informando quais são os valores de entrada da função, quais são os valores de saída, quais as relações que esses valores que entram e saem da função e as transformações pela função realizadas.



Documentação

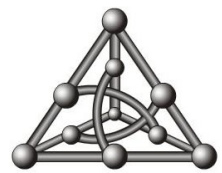
Segundo o professor Paulo Feofiloff, uma boa documentação de um programa significa:

- inserir comentários apropriados no código de modo a explicar *o que* cada uma das funções que compõem o programa faz;
- a documentação de uma função é um pequeno manual que dá instruções precisas e completas sobre o uso da função;
- uma boa documentação não se preocupa em explicar *como* uma função faz o que faz, mas sim *o que* ela faz de fato, informando quais são os valores de entrada da função, quais são os valores de saída, quais as relações que esses valores que entram e saem da função e as transformações pela função realizadas.



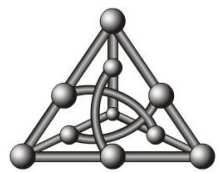
Documentação

```c
#include <stdio.h>

/* Este programa faz a adição de dois números inteiros
   fixos e mostra o resultado da operação na saída. */
int main()
{
   int num1, num2, soma;

   num1 = 25;
   num2 = 30;
   soma = num1 + num2;
   printf("A soma de %d e %d é %d\n", num1, num2, soma);

   return 0;
}
```



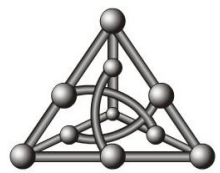
Entrada e saída

```
#include <stdio.h>

/* Recebe dois números inteiros e imprime sua soma */
int main()
{
   int num1, num2, soma;

   printf("Informe um número: \n");
   scanf("%d", &num1);
   printf("Informe outro número: \n");
   scanf("%d", &num2);

   soma = num1 + num2;

   printf("A soma de %d mais %d é %d\n", num1, num2, soma);

   return 0;
}
```



Entrada e saída

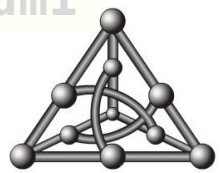
```c
#include <stdio.h>

/* Recebe dois números inteiros e imprime sua soma */
int main()
{
   int num1, num2, soma;

   printf("Informe um número: \n");
   scanf("%d", &num1);
   printf("Informe outro número: \n");
   scanf("%d", &num2);

   soma = num1 + num2;

   printf("A soma de %d mais %d é %d\n", num1, num2, soma);

   return 0;
}
```



Entrada e saída

```
printf("Informe um número: ");
scanf("%d", &num1);
```

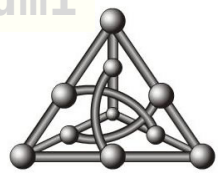
- a **função de leitura** `scanf` tem dois argumentos: uma cadeia de caracteres de formatação `"%d"` e uma variável `num1` correspondente a esse formato
- diferentemente da função `printf`, uma variável do tipo inteiro que é um argumento da função `scanf` deve *sempre* vir precedida com o símbolo `&`
- na instrução, a cadeia de caracteres de formatação `"%d"` indica que o valor informado pelo usuário será convertido para um inteiro
- este valor será então armazenado na variável do tipo inteiro `num1`



Entrada e saída

```
printf("Informe um número: ");
scanf("%d", &num1);
```

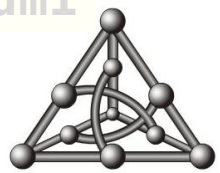
- a **função de leitura** `scanf` tem dois argumentos: uma cadeia de caracteres de formatação `"%d"` e uma variável `num1` correspondente a esse formato
- diferentemente da função `printf`, uma variável do tipo inteiro que é um argumento da função `scanf` deve *sempre* vir precedida com o símbolo `&`
- na instrução, a cadeia de caracteres de formatação `"%d"` indica que o valor informado pelo usuário será convertido para um inteiro
- este valor será então armazenado na variável do tipo inteiro `num1`



Entrada e saída

```
printf("Informe um número: ");
scanf("%d", &num1);
```

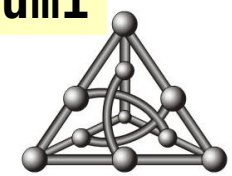
- a **função de leitura** `scanf` tem dois argumentos: uma cadeia de caracteres de formatação `"%d"` e uma variável `num1` correspondente a esse formato
- diferentemente da função `printf`, uma variável do tipo inteiro que é um argumento da função `scanf` deve *sempre* vir precedida com o símbolo `&`
- na instrução, a cadeia de caracteres de formatação `"%d"` indica que o valor informado pelo usuário será convertido para um inteiro
- este valor será então armazenado na variável do tipo inteiro `num1`



Entrada e saída

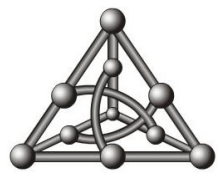
```
printf("Informe um número: ");
scanf("%d", &num1);
```

- a **função de leitura** `scanf` tem dois argumentos: uma cadeia de caracteres de formatação `"%d"` e uma variável `num1` correspondente a esse formato
- diferentemente da função `printf`, uma variável do tipo inteiro que é um argumento da função `scanf` deve *sempre* vir precedida com o símbolo `&`
- na instrução, a cadeia de caracteres de formatação `"%d"` indica que o valor informado pelo usuário será convertido para um inteiro
- este valor será então armazenado na variável do tipo inteiro `num1`



Simulação passo a passo da execução de programas

```
01  #include <stdio.h>
02
03  int main()
04  {
05    int num1, num2, produto;
06
07    printf("Informe um número: ");
08    scanf("%d", &num1);
09    printf("Informe outro número: ");
10    scanf("%d", &num2);
11    produto = num1 * num2;
12    printf("O produto de %d por %d é %d\n", num1, num2, produto);
13
14    return 0;
15  }
```



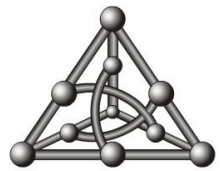
Simulação passo a passo da execução de programas

programa

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

memoria

entrada/saida



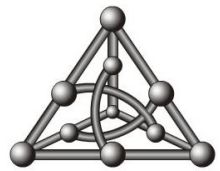
Simulação passo a passo da execução de programas

programa

```
02
```

memoria

entrada/saida



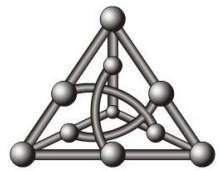
Simulação passo a passo da execução de programas

programa

```
03 int main(void)
```

memoria

entrada/saida



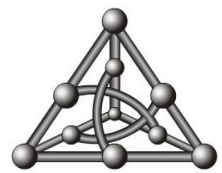
Simulação passo a passo da execução de programas

programa

```
04 {
```

memoria

entrada/saida

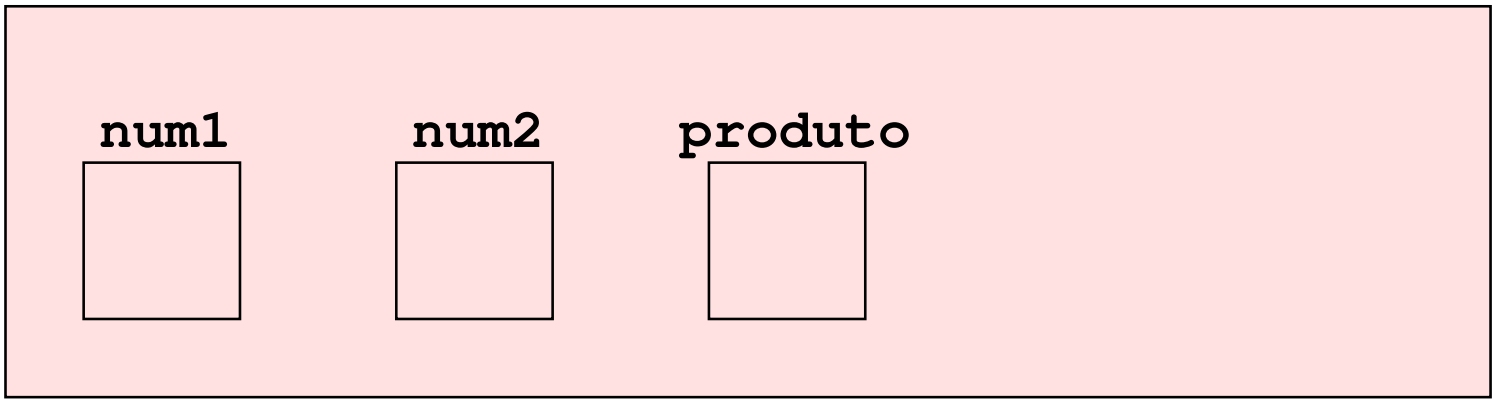


Simulação passo a passo da execução de programas

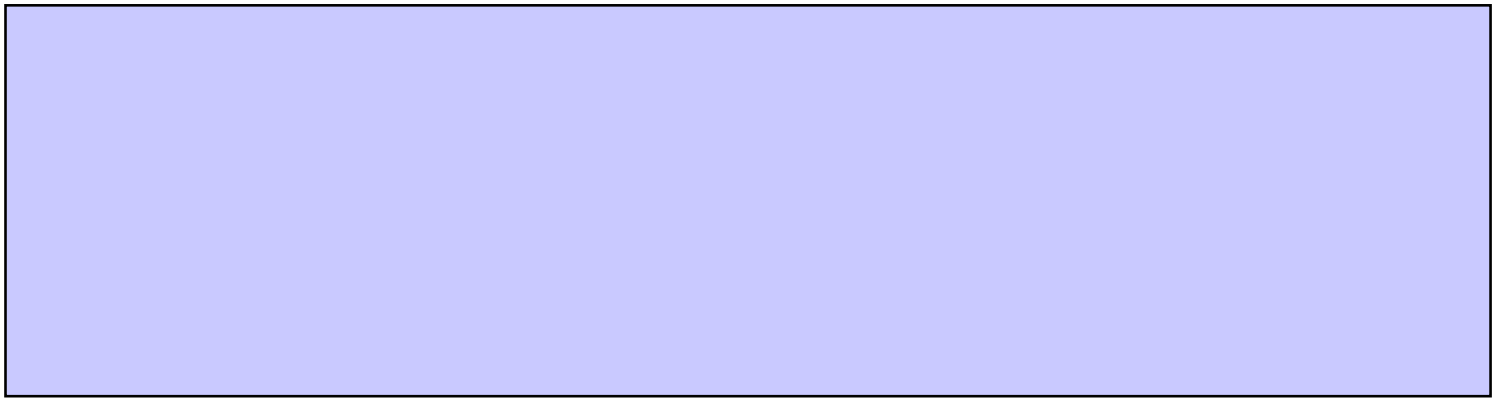
programa

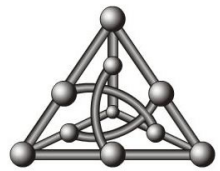
```
05      int num1, num2,  produto;
```

memoria



entrada/saida



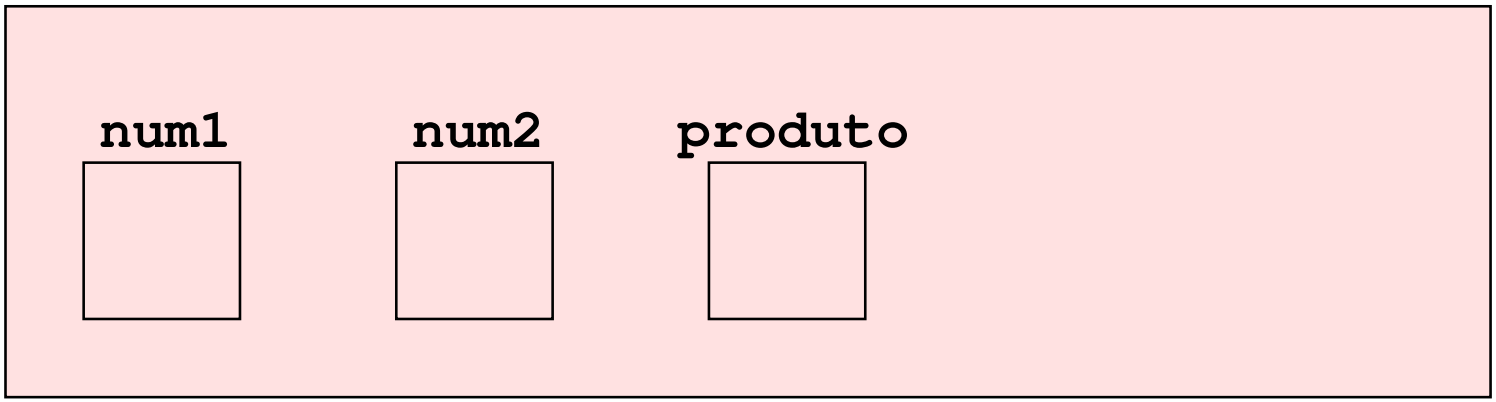


Simulação passo a passo da execução de programas

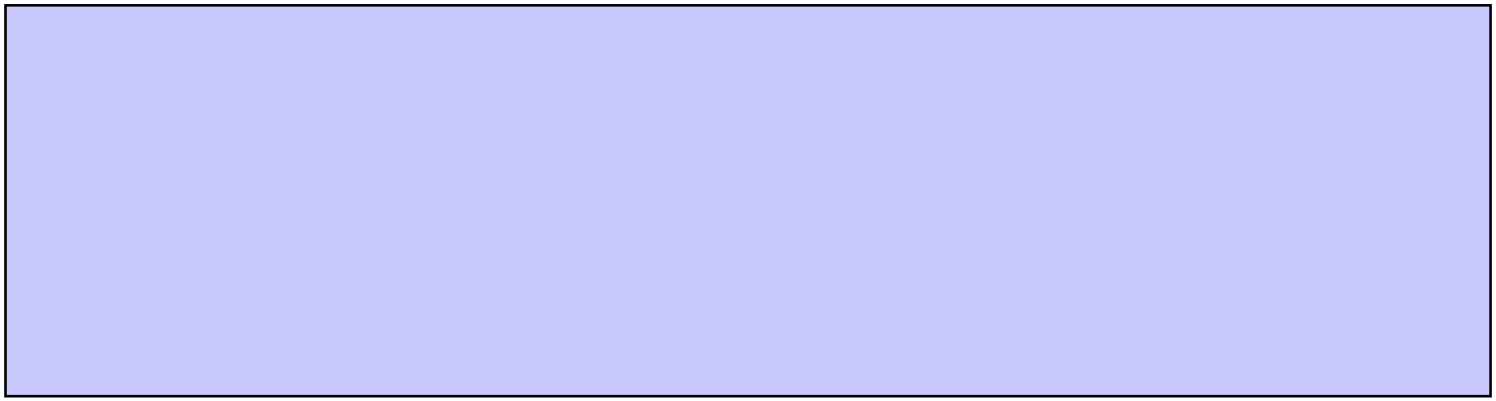
programa

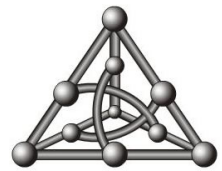
```
06
```

memoria



entrada/saida



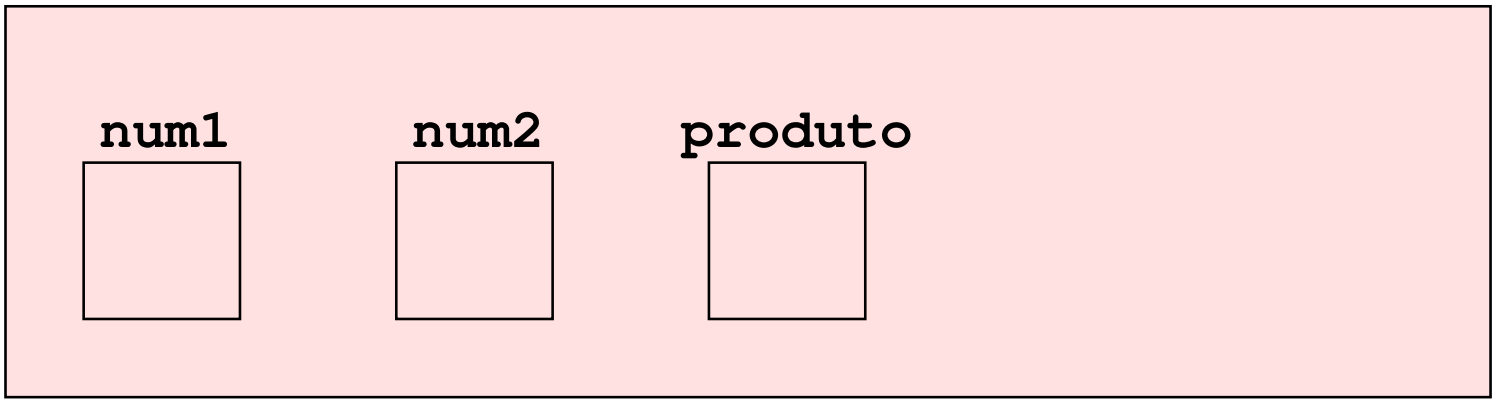


Simulação passo a passo da execução de programas

programa

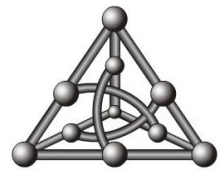
```
07      printf("Informe um numero: ");
```

memoria



entrada/saida

```
Informe um numero:
```

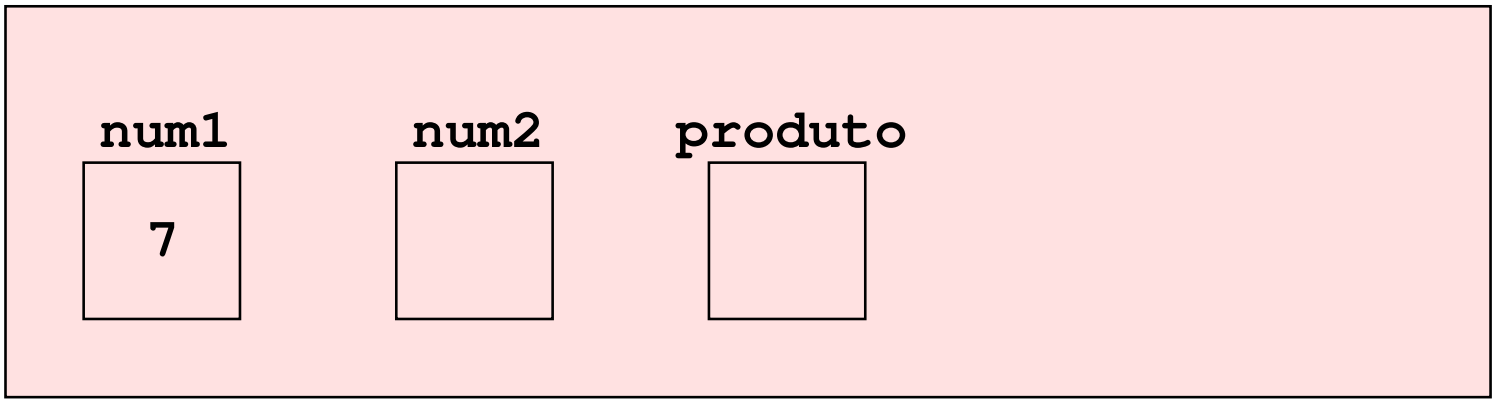


Simulação passo a passo da execução de programas

programa

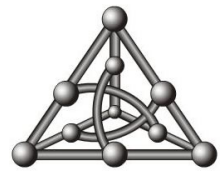
```
08      scanf("%d", &num1);
```

memoria



entrada/saida

```
Informe um numero:  7
```

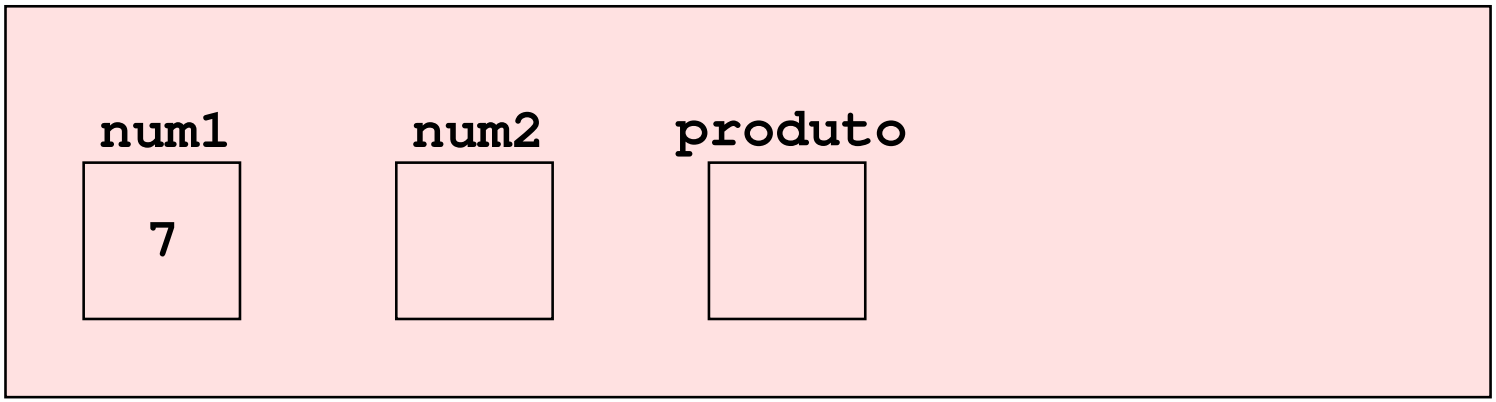


Simulação passo a passo da execução de programas

programa

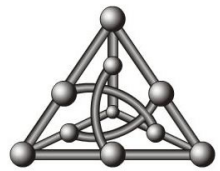
```
09      printf("Informe outro numero: ");
```

memoria



entrada/saida

```
Informe um numero:  7
Informe outro numero:
```

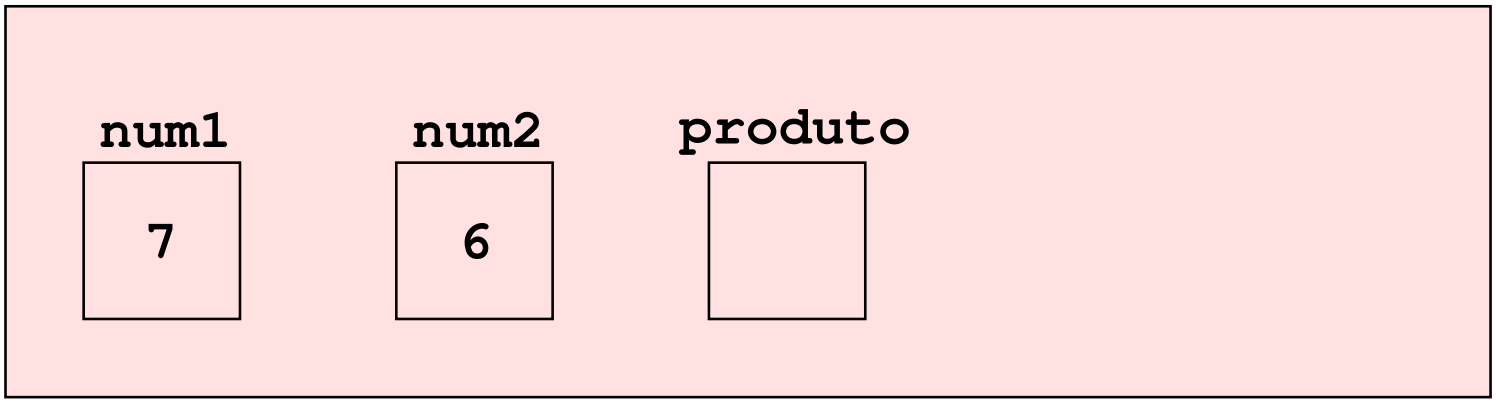


Simulação passo a passo da execução de programas

programa

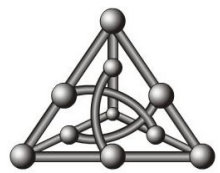
```
10      scanf("%d", &num2);
```

memoria



entrada/saida

```
Informe um numero:  7
Informe outro numero:  6
```

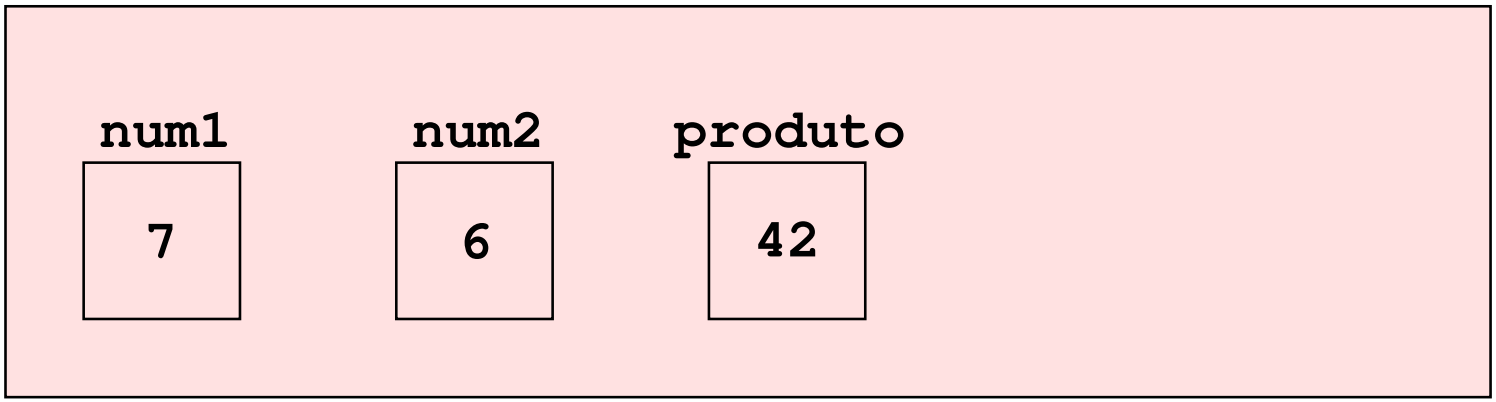


Simulação passo a passo da execução de programas

programa

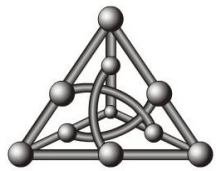
```
11      produto = num1 * num2;
```

memoria



entrada/saida

```
Informe um numero:  7
Informe outro numero:  6
```

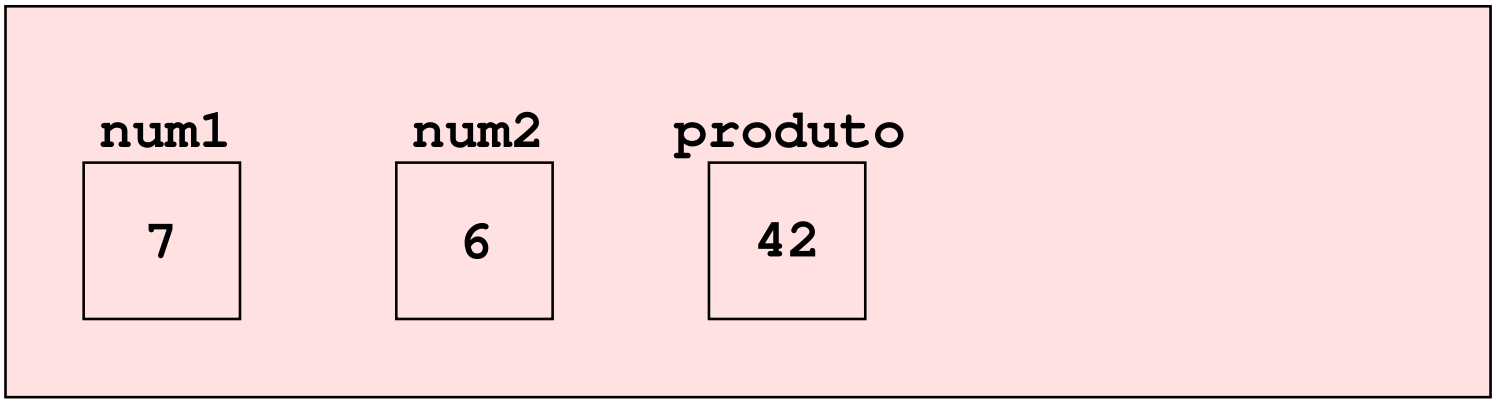


Simulação passo a passo da execução de programas

programa

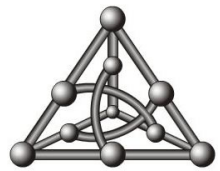
```
12      print("O produto de %d por %d ...
```

memoria



entrada/saida

```
Informe um numero:  7
Informe outro numero:  6
O produto de 7 por 6 e 42
```



Simulação passo a passo da execução de programas

programa

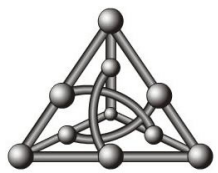
```
13
```

memoria

num1	num2	produto
7	6	42

entrada/saida

```
Informe um numero:  7
Informe outro numero:  6
O produto de 7 por 6 e 42
```

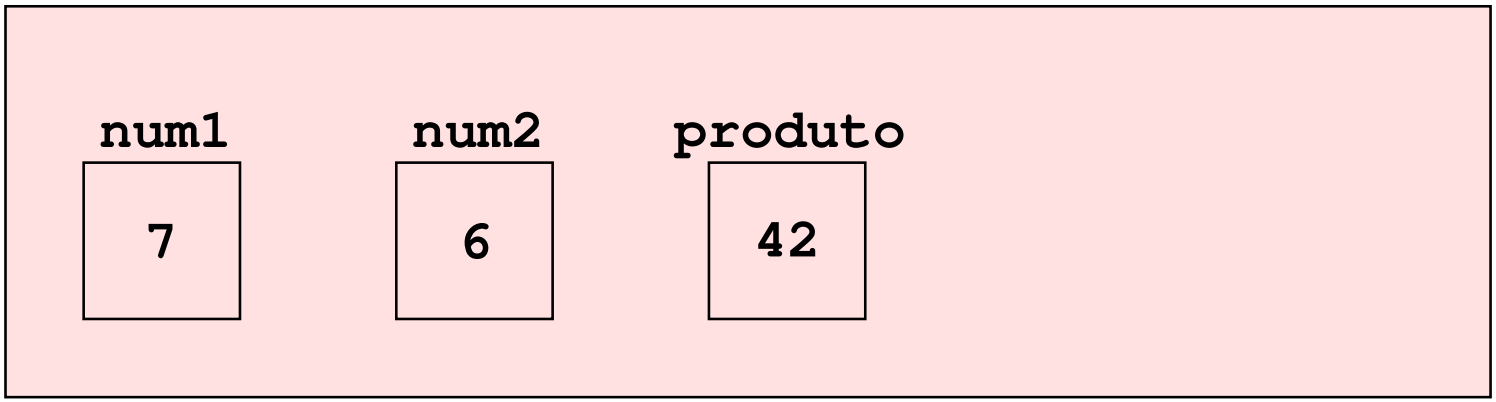


Simulação passo a passo da execução de programas

programa

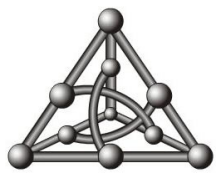
```
14      return 0;
```

memoria



entrada/saida

```
Informe um numero:  7
Informe outro numero:  6
O produto de 7 por 6 e 42
```

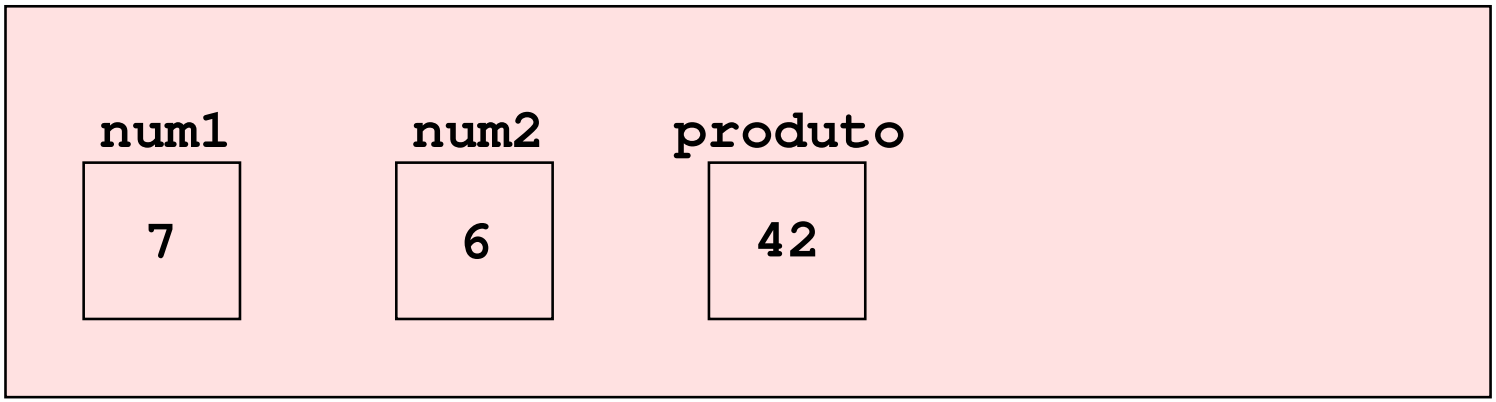


Simulação passo a passo da execução de programas

programa

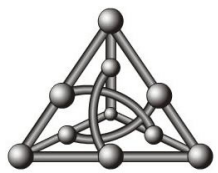
```
15          }
```

memoria



entrada/saida

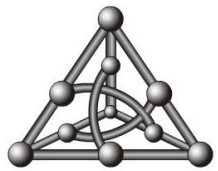
```
Informe um numero:  7
Informe outro numero:  6
O produto de 7 por 6 e 42
```



Exercícios

1. Escreva uma programa na linguagem C que escreva a seguinte mensagem na saída padrão:

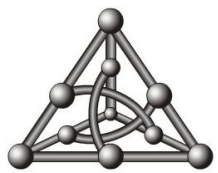
```
a) Comentários na linguagem C iniciam com /* e terminam com */
b) Letras minúsculas e maiúsculas são diferentes na linguagem C
c) A palavra-chave main indica o início do programa na linguagem C
d) Os símbolos { e } envolvem um bloco de comandos na linguagem C
e) Todos as instruções na linguagem C devem terminar com um ;
```



Exercícios

2. Qual é a saída esperada do programa a seguir?

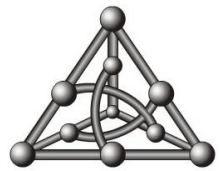
```
#include <stdio.h>

int main()
{

   printf("Alô! ");
   printf("Alô! ");
   printf("Tem alguém aí?");
   printf("\n");

   return 0;
}
```



Exercícios

3. Escreva um programa na linguagem C que subtraia 14 de 73 e mostre o resultado na saída padrão com uma mensagem apropriada. Faça a simulação passo a passo da execução do programa.
4. Verifique se o programa abaixo está correto. Em caso negativo, liste os erros de digitação que você encontrou.

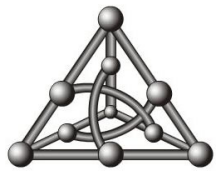
```
include <stdio.h>
/* computa a soma de dois números
Int main()
{
   int resultado;
   resultado = 13 + 22 - 7
   printf("O resultado da operação é %d\n"   resultado);
   return 0;
}
```



Exercícios

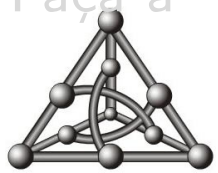
3. Escreva um programa na linguagem C que subtraia 14 de 73 e mostre o resultado na saída padrão com uma mensagem apropriada. Faça a simulação passo a passo da execução do programa.
4. Verifique se o programa abaixo está correto. Em caso negativo, liste os erros de digitação que você encontrou.

```
include <stdio.h>
/* computa a soma de dois números
Int main()
{
   int resultado;
   resultado = 13 + 22 - 7
   printf("O resultado da operação é %d\n"   resultado);
   return 0;
}
```



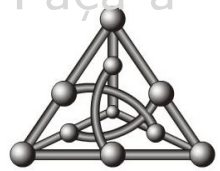
Exercícios

5. Escreva um programa que leia três números inteiros a, b e c, calcule $a * b + c$ e mostre o resultado na saída padrão para o(a) usuário(a). Faça a simulação passo a passo da execução do programa.
6. Escreva um programa que leia um número inteiro e mostre o seu quadrado e seu cubo. Por exemplo, se o número de entrada é 3, a saída deve ser 9 e 27. Faça a simulação passo a passo da execução do programa.
7. Escreva um programa que leia três números inteiros e mostre como resultado a soma desses três números e também a multiplicação desses três números. Faça a simulação passo a passo da execução do programa.
8. Escreva um programa que leia um número inteiro e mostre o resultado do quociente da divisão desse número por 2 e por 3. Faça a simulação passo a passo da execução do programa.



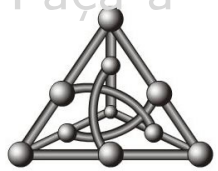
Exercícios

5. Escreva um programa que leia três números inteiros a, b e c, calcule $a * b + c$ e mostre o resultado na saída padrão para o(a) usuário(a). Faça a simulação passo a passo da execução do programa.
6. Escreva um programa que leia um número inteiro e mostre o seu quadrado e seu cubo. Por exemplo, se o número de entrada é 3, a saída deve ser 9 e 27. Faça a simulação passo a passo da execução do programa.
7. Escreva um programa que leia três números inteiros e mostre como resultado a soma desses três números e também a multiplicação desses três números. Faça a simulação passo a passo da execução do programa.
8. Escreva um programa que leia um número inteiro e mostre o resultado do quociente da divisão desse número por 2 e por 3. Faça a simulação passo a passo da execução do programa.



Exercícios

5. Escreva um programa que leia três números inteiros a, b e c, calcule $a * b + c$ e mostre o resultado na saída padrão para o(a) usuário(a). Faça a simulação passo a passo da execução do programa.
6. Escreva um programa que leia um número inteiro e mostre o seu quadrado e seu cubo. Por exemplo, se o número de entrada é 3, a saída deve ser 9 e 27. Faça a simulação passo a passo da execução do programa.
7. Escreva um programa que leia três números inteiros e mostre como resultado a soma desses três números e também a multiplicação desses três números. Faça a simulação passo a passo da execução do programa.
8. Escreva um programa que leia um número inteiro e mostre o resultado do quociente da divisão desse número por 2 e por 3. Faça a simulação passo a passo da execução do programa.



Exercícios

5. Escreva um programa que leia três números inteiros a, b e c, calcule $a * b + c$ e mostre o resultado na saída padrão para o(a) usuário(a). Faça a simulação passo a passo da execução do programa.
6. Escreva um programa que leia um número inteiro e mostre o seu quadrado e seu cubo. Por exemplo, se o número de entrada é 3, a saída deve ser 9 e 27. Faça a simulação passo a passo da execução do programa.
7. Escreva um programa que leia três números inteiros e mostre como resultado a soma desses três números e também a multiplicação desses três números. Faça a simulação passo a passo da execução do programa.
8. Escreva um programa que leia um número inteiro e mostre o resultado do quociente da divisão desse número por 2 e por 3. Faça a simulação passo a passo da execução do programa.