BCC 201 - Introdução à Programação

Variáveis, Comandos de Atribuição e Comando de Entrada e Saída

Guillermo Cámara-Chávez
UFOP

Estrutura Básica de um programa C I

```
< diretivas do pre-processador >
< declaracoes globais >;
int main()
{
    < declaracoes locais >;  /* comentario */
    < instrucoes >;
    return 0;
}
< outras funcoes >
```

Estrutura Básica de um programa C II

```cpp
/* Prog. C++: Bom dia */
#include <iostream>
using namespace std;
int main()
{
    cout<<"Bom Dia!!";
    return 0;
}
```

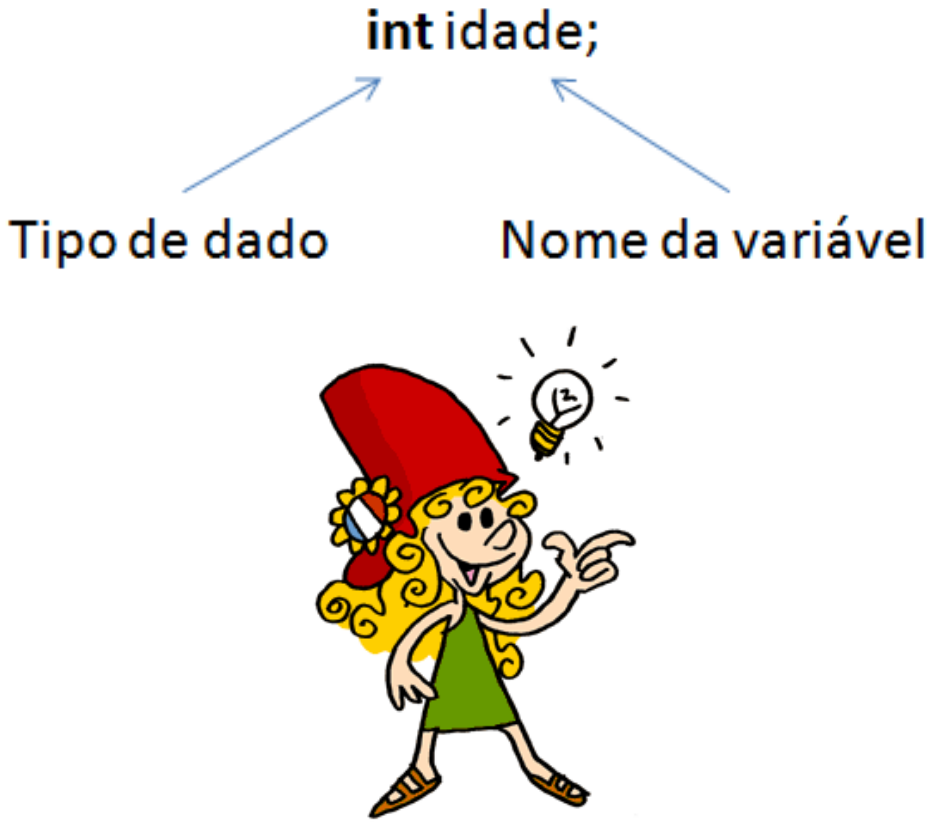
```c
/* Prog. C: Bom dia */
#include <stdio.h>

int main()
{
    printf("Bom Dia!!");
    return 0;
}
```

- *main()* é única, determina o inicio do programa.
- O comando *return* informa ao sistema operacional se o programa funcionou corretamente ou não.

Variáveis

- São locais onde armazenamos valores na memória.
- Toda variável é caracterizada por:
  - um nome: a identifica em um programa,
  - um tipo: determina o que pode ser armazenado naquela variável

Declarando uma variável I



Tipos de variáveis I

- Inteiras: utilizadas para armazenar valores inteiros

Tipo	**Tamanho**	**Valores possíveis**
(short) (signed) int	2 Bytes	-32.768 a 32.767
(short) unsigned int	2 Bytes	0 a 65.535
(signed) long int	4 Bytes	-2.147.483.648 a 2.147.483.647
unsigned long int	4 Bytes	0 a 4.294.967.295

Tipos de variáveis II

- Variáveis de tipo caracter
  - Utilizadas para armazenar letras e outro símbolos existentes em textos
  - São, na verdade, variáveis inteiras que armazenam um número associado ao símbolo. A principal tabela de símbolos utilizada pelo computadores é a tabela ASCII (*American Standard Code for Information Interchange*), mas existem outras (EBCDIC, Unicode, etc.)

Tipo	**Tamanho**	**Valores possíveis**
(signed) char	1 Byte	-128 a 127
unsigned char	1 Byte	0 a 256

Tipos de variáveis III

- Variáveis de tipo ponto flutuante
  - Armazenam valores reais, da seguinte forma

$$(-1)^{sinal} * mantissa * 2^{expoente}$$

    Ex: $0.5 = (-1)^0 * 1 * 2^{-1}$.
  - Para o programador, funciona como se ele armazenasse números na forma decimal.
  - Possui problema de precisão (arredondamento).

Tipo	**Tamanho**	**Valores possíveis**
float	4 Bytes	$\pm 3,4E-38$ a $\pm 3,4E+38$
long float	8 Bytes	$\pm 1,7E-308$ a $\pm 1,7E+308$
double	8 Bytes	$\pm 1,7E-308$ a $\pm 1,7E+308$

Obtendo o tamanho de um tipo I

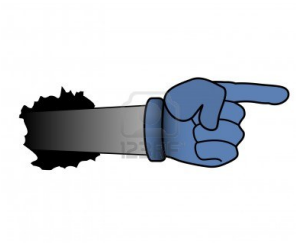
- O comando `sizeof`(tipo) retorna o tamanho, em *bytes*, de um determinado tipo. (Um *byte* corresponde a 8 *bits*).
- Ex. `cout<<sizeof(int);`
- Escreve 4 na tela.

Obtendo o tamanho de um tipo II

Mostrar em *Bytes* o tamanho ocupado por variáveis do tipo inteiro, real e caracter.

```
#include <stdio.h>
int main(){
    cout<<"int      : " << sizeof(int) << endl;
    cout<<"long     : " << sizeof(long int) << endl;
    cout<<"float    : " << sizeof(float) << endl;
    cout<<"double   : " << sizeof(double) << endl;
    cout<<"char     : " << sizeof(char) << endl;
    return 0;
}
```

Regras para nomes de variáveis em C++ I



- Deve começar com uma letra (maiúscula ou minúscula) ou subscrito (_ *underscore*).
- **Nunca** pode começar com um número.
- Pode conter letras maiúsculas, minúsculas, número e subscrito
- Não pode-se utilizar `{ ( + - / \ ; . , ?` como parte do nome de uma variável.

Regras para nomes de variáveis em C++ II

- C/C++ são uma linguagem *case-sensitive*, ou seja, faz diferença entre nomes com letras maiúsculas e nomes com letras minúsculas: `Peso` e `peso` são diferentes.
- Costuma-se usar maiúsculas e minúsculas para separar palavras `PesoDoCarro`
- Identificadores devem ser únicos no mesmo escopo (não podem haver variáveis com mesmo identificador dentro do mesmo bloco).

Regras para nomes de variáveis em C++ III

- As seguintes palavras já tem um significado na linguagem C e por esse motivo não podem ser utilizadas como nome de variáveis:

auto	*double*	*int*	*struct*	*break*
enum	*register*	*typedef*	*char*	*extern*
return	*union*	*const*	*float*	*short*
unsigned	*continue*	*for*	*signed*	*void*
default	*goto*	*sizeof*	*volatile*	*do*

Regras para nomes de variáveis em C++ IV

- Quais dos nomes a seguir são nomes corretos de variáveis? Se não forem corretos, porque não são?

3ab	a3b	fim	int
`\meu`	`_A`	`n_a_o`	papel-branco
`a*`	`c++`	`*nova_variavel`	

Regras para nomes de variáveis em C++ V

- Exemplo 1:

```
/* Exemplo de programa em C++ */
//  Arquivo de cabecalho (header)
#include <iostream>
using namespace std;
int main()
{
    int contador;  // declaracoes simples
    float PrecoDoQuilo;
    double TaxaDeCambio;
    char LetraDigitada;
    // Pode colocar mais de uma variavel na
    // na mesma linha
    int IdadeManoel, IdadeJoao, IdadeMaria;
    double  TaxaDoDolar,     // Tambem pode trocar
            TaxaDoMarco,     // de linha no meio
            TaxaDoPeso,
            TaxaDoFranco;
.......
}
```

Regras para nomes de variáveis em C++ VI

- Exemplo 2: inicialização de variáveis

```
// Exemplo de programa em C++
//  Arquivo de cabecalho (header)
#include <iostream>
int main()
{
    // declara e inicializa com Zero
    int NroDeHoras = 0;
    // declara e inicializa com 10.53
    float PrecoDoQuilo = 10.53;
    double  TaxaDoDolar = 1.8,
            TaxaDoMarco = 1.956,
            TaxaDoPeso = 1.75,
            TaxaDoFranco = 0.2;


.......
    return 0;
}
```

Constantes I

- Constantes são valores previamente determinados e que, por algum motivo, devem aparecer dentro de um programa (veremos adiante onde elas podem ser usadas).
- Assim como as variáveis, as constantes também possuem um tipo. Os tipos permitidos são exatamente os mesmos das variáveis, mais o tipo *string*, que corresponde a uma seqüência de caracteres.
- Exemplos de constantes: 85, 0.10, 'c', "Meu primeiro programa"

Exemplo de declaração de constantes I

```
// Nao se coloca ponto-e-virgula apos o valor
#define LARGURA_MAXIMA 50
#define NRO_DE_DIAS_DA_SEMANA 7
#define NRO_DE_HORAS_DO_DIA 24
#define VALOR_DE_PI  3.1415

int main ()
{
    int TotalDeHoras;
    const int r = 100;

    TotalDeHoras = 10 * NRO_DE_DIAS_DA_SEMANA *
                   NRO_DE_HORAS_DO_DIA;
.......
    return 0;
}
```

Escrevendo o conteúdo de uma variável na tela em C++ I

- C++ usa o conceito de *streams* (fluxos) para executar operações de entrada e saída
- Uma *stream* é um objeto no qual um programa pode inserir dados ou do qual ele pode extrair dados.
- Para se utilizar streams, é necessário incluir a biblioteca `iostream`.

Escrevendo o conteúdo de uma variável na tela em C++ II

- Por default, a saída padrão envia dados para a tela e o objeto *stream* é identificado como `cout`.
- `cout` é usado em conjunto com o operador de inserção (<<).
- Exemplo: `cout << x << endl;`

Formatação de saída I

- *I/O manipulators* são a forma mais comum de controlar a formatação de saída. Usar a biblioteca `<iomanip>`
- Alguns métodos para manipular a formatação de saída:

Método	Descrição
`endl`	escreve uma nova linha
`setw(n)`	define o tamanho da saída. Só afeta ao elemento que vem a continuação
`width(n)`	igual que `setw(n)`
`left`	justifica à esquerda, so pode ser usado depois de `setw`
`right`	justifica à direita, so pode ser usado depois de `setw`

Formatação de saída II

- Exemplo 1:

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    float x = 25.65749;
    cout << setw(10) << x;

    return 0;
}
```

imprime ␣␣25.65749

Formatação de saída III

Método	Descrição
`setfill(ch)`	usado depois de `setw`, preenche os espaços com o caracter definido em `ch`
`fixed`	mostra os decimais de um núm. real, por *default* são 6 decimais
`setprecision(n)`	define o número de decimais que serão mostrados. Deve ser usado junto com `fixed`. De não ser assim conta o número total de dígitos (inteiros e decimais).

Formatação de saída IV

- Exemplo 2:

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    float x = 25.65749;
    cout << setfill('0')<< setw(11) << x;
    return 0;
}
```

imprime 00025.65749

Formatação de saída V

- Exemplo 3:

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    float x = 49325.65749;
    cout << setprecision(4) << x;
    return 0;
}
```

imprime 4.933e+004

Formatação de saída VI

- Exemplo 4:

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    float x = 49325.65749;
    cout << fixed << setprecision(4) << x;
    return 0;
}
```

imprime 49325.6575

Formatação de saída VII

```
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    const float A = 0.1;
    const float um  = 1.0;
    const float big  = 1234567890.0;
    const float B = 4567.87683;

    cout<<"A. "<<A<<", "<<um<<", "<<big<<endl;
    cout<<"B. "<<setprecision(5)<<B<<endl;
    cout<<"C. "<<fixed<<A <<", "<<um<<", "<<big<<endl;
    cout<<"D. "<<fixed<<setprecision(3)<<A<<", "
        <<um<<", "<<big<<endl;
    cout<<"E. "<<setprecision(20)<<A<<endl;
    cout<<"F. "<<setw(8)<<setfill('*')<<34<<45<<endl;
    cout<<"G. "<<setw(8)<<34<<setw(8)<<45<<endl;
    system("pause");
    return 0;
}
```

Formatação de saída VIII

```
Mostra na tela
```

A. 0.1, 1, 1.23457e+009
B. 4567.9
C. 0.10000, 1.00000, 1234567936.00000
D. 0.100, 1.000, 1234567936.000
E. 0.100000001490116120000
F. ******3445
G. ******34******45

A função cin (C++) I

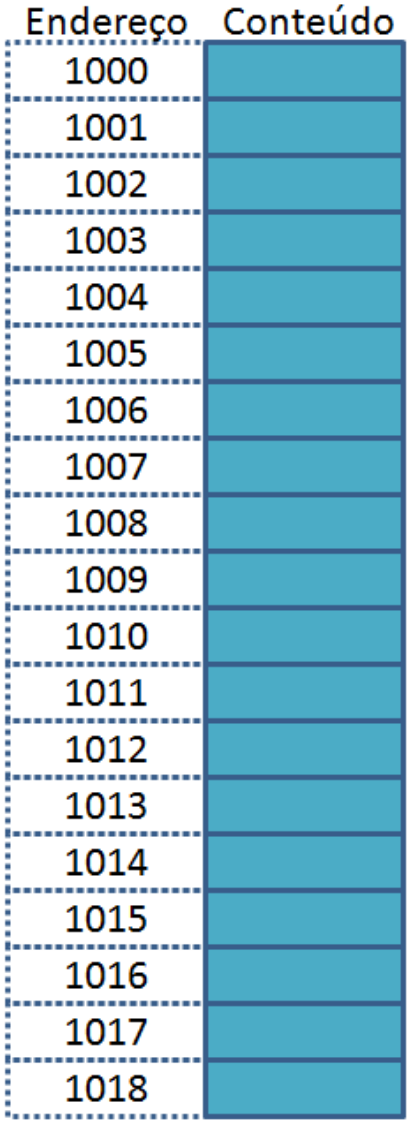
- O operador >> sobrecarregado executa a entrada com streams em C++.
- O comando `cin` é utilizado para aquisição de dados

A função cin (C++) II

```
#include <iostream>
int main(){
    int n;
    cout  << "Digite um numero: ";
    cin >> n;
    cout << "O valor digitado foi " << n << endl;
    return 0;
}
```

A função cin (C++) III

Memória



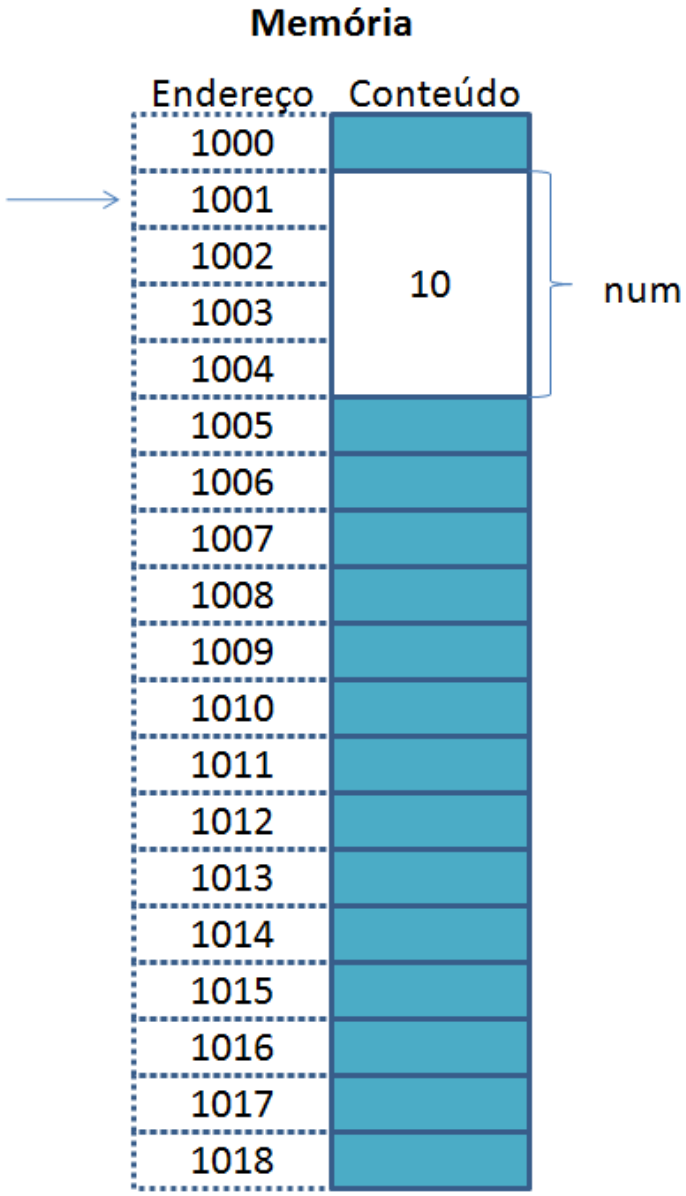
Código

```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
cout << "Inserir nota";
cin >> nota;
```

Tela

Ação

A função cin (C++) IV



Código

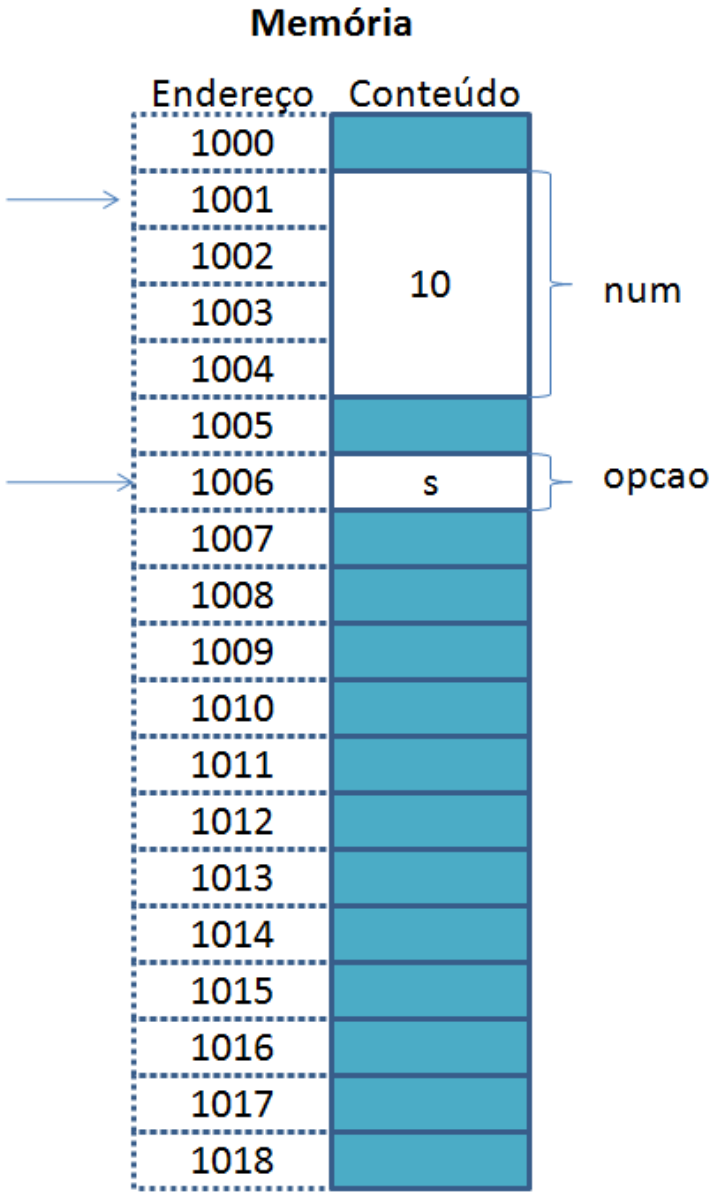
```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
cout << "Inserir nota";
cin >> nota;
```

Tela

Ação

Declara e separa memória para a variável ***num*** do tipo inteiro
Atribui o valor 10

A função cin (C++) V



Código

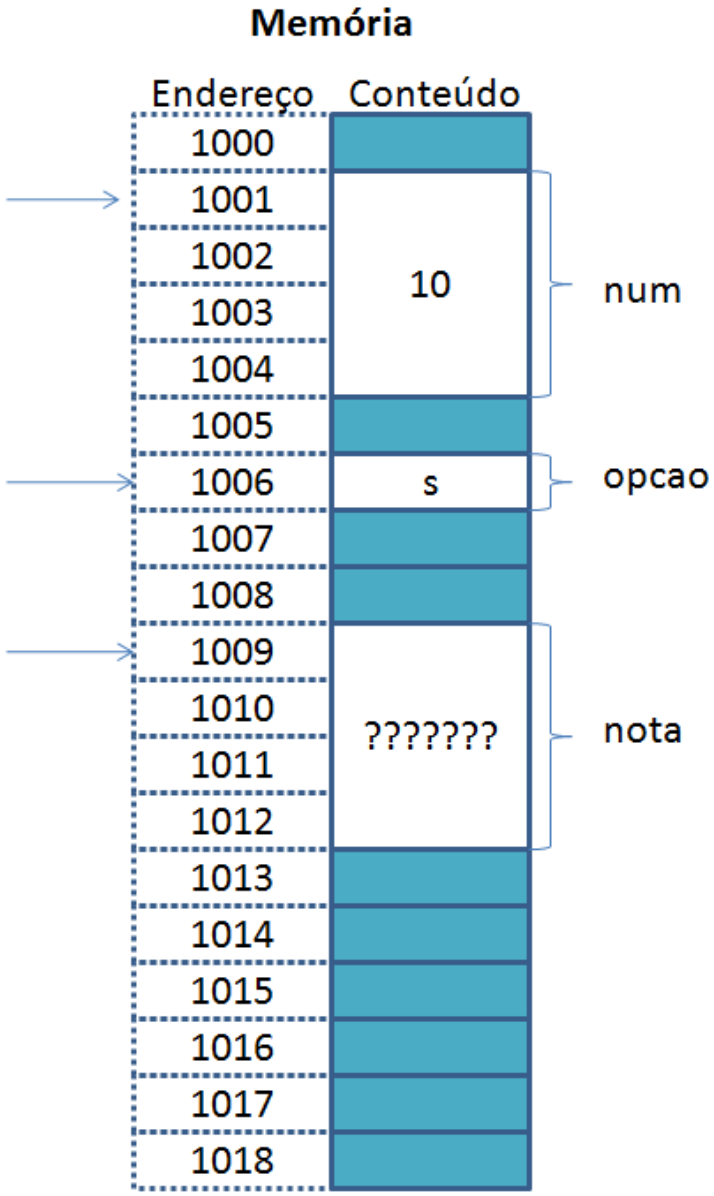
```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
cout << "Inserir nota";
cin >> nota;
```

Tela

Ação

Declara e separa memória para a variável ***opcao*** do tipo caracter
Atribui o valor 's'

A função cin (C++) VI



Código

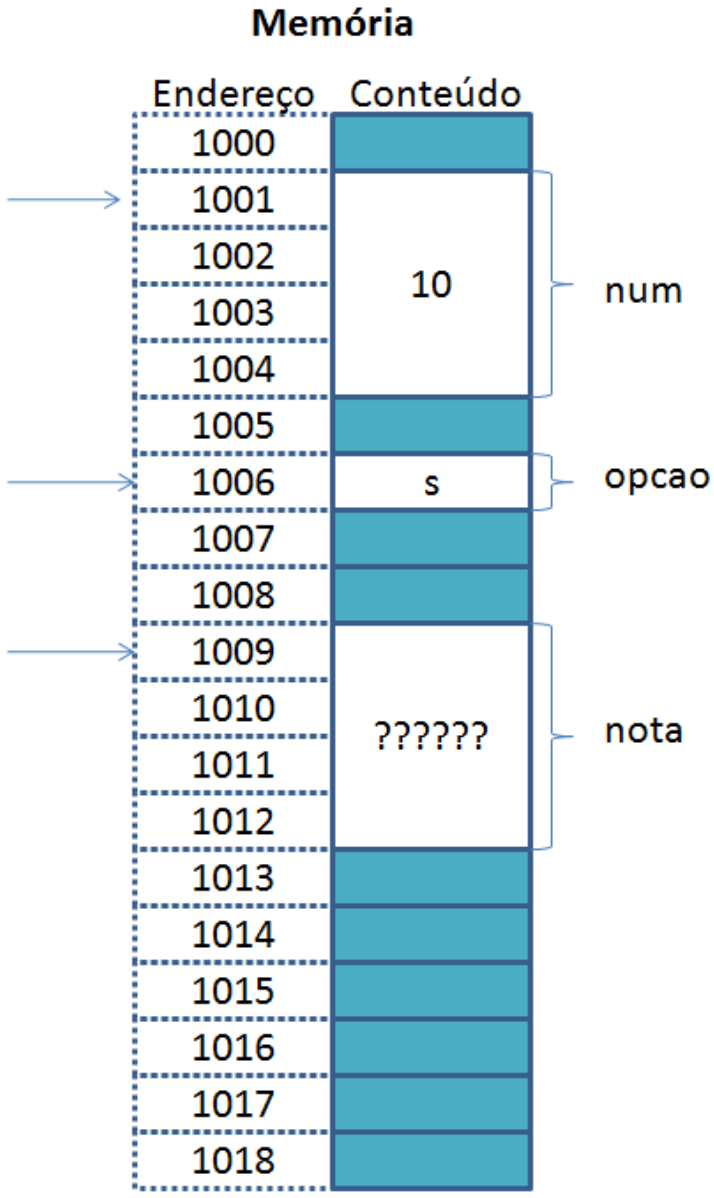
```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
cout << "Inserir nota";
cin >> nota;
```

Tela

Ação

Declara e separa memória para a variável ***nota*** do tipo real de precisão simples

A função cin (C++) VII



Código

```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
cout << "Inserir nota";
cin >> nota;
```

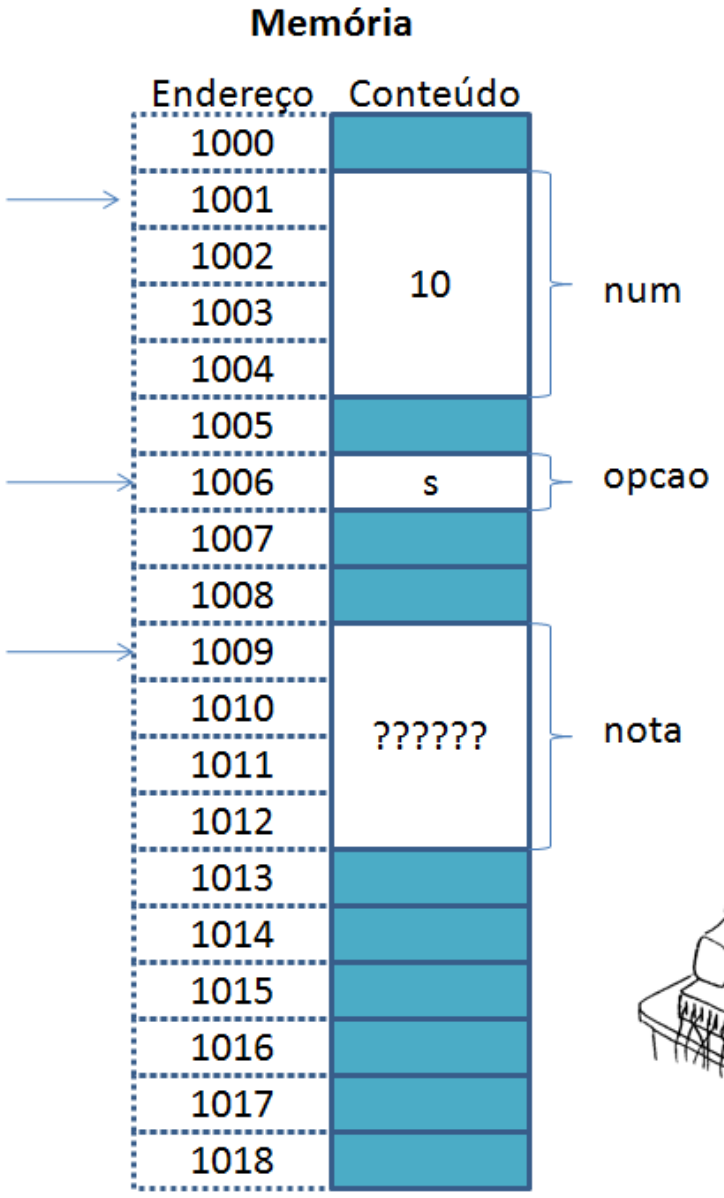
Tela

Inserir nota

Ação

Mostra na tela a mensagem : "Inserir nota"

A função cin (C++) VIII



Código

```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
cout << "Inserir nota";
cin >> nota;
```

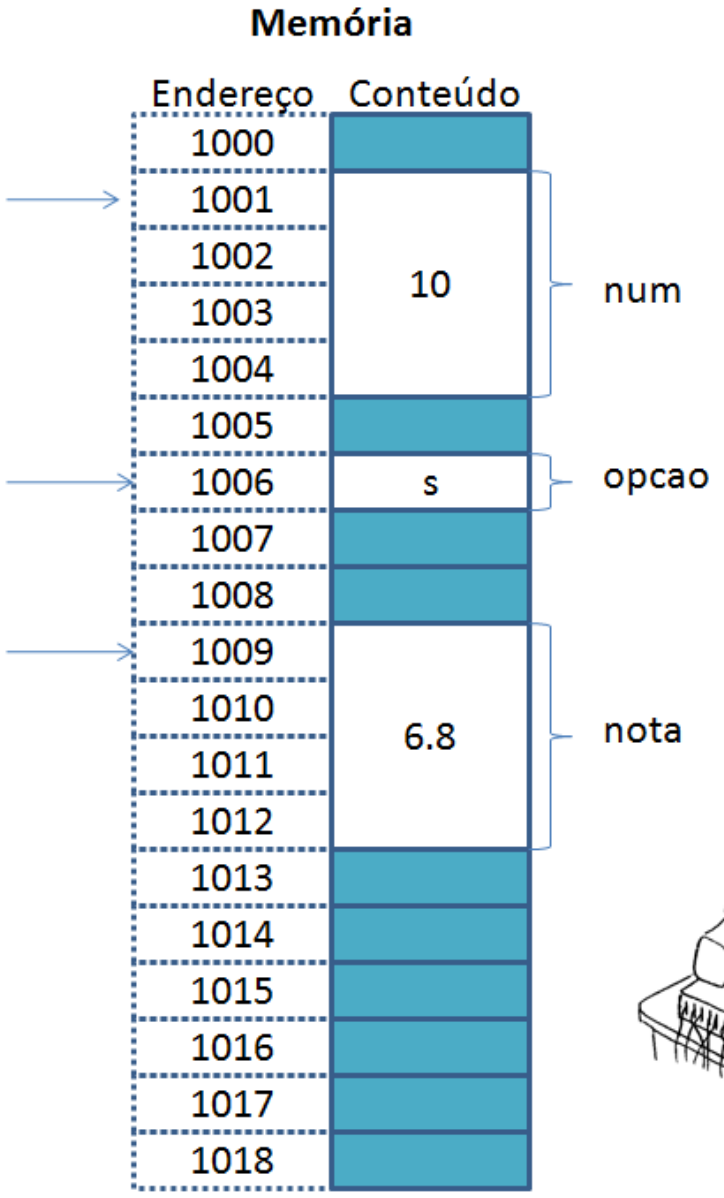
Tela

Inserir nota _

Ação

Operação de leitura, um valor número de tipo real será inserido no endereço de memória **1009**

A função cin (C++) IX



Código

```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
cout << "Inserir nota";
cin >> nota;
```

Tela

Inserir nota 6.8

Ação

Usuário digita um número

O endereço de uma variável I

- Toda variável tem um **endereço de memória** associada a ela.
- Esse endereço é o **local** onde essa variável é armazenada no sistema (como se fosse o endereço de uma casa, o local onde as pessoas são "armazenadas").

Endereço	Conteúdo	Variável
⋮	⋮	⋮
1001	???	
1002	2450	i
1003	???	
1004	225.345	f
1005	11331	j

O endereço de uma variável II

- Normalmente, o endereço das variáveis não são conhecidos quando o programa é escrito.
- O endereço é dependente do sistema computacional e também da implementação do compilador C que está sendo usado.
- O endereço de uma mesma variável pode mudar entre diferentes execuções de um mesmo programa C usando uma mesma máquina.

Leitura de dados I

```
#include <iostream>
using namespace std;
int main(){
    char car;
    cout << "Digite um caracter: ";
    cin >> car;
    cout << "O caracter digitado foi: " << car <<"\n";
    return 0;
}
```

Leitura de dados II

Crie um programa que permita inserir o nome e a idade de um aluno

Leitura de dados III

```
#include<iostream>
#include<string>
using namespace std;
int main(){
    string nome;
    int idade;
    cout << "\n Inserir nome: ";
    cin >> nome;
    cout << "\n Inserir idade: ";
    cin >> idade;
    cout << nome << " tem " << idade << anos;
    return 0;
}
```

Leitura de dados IV

Crie um programa que permita inserir o nome completo (nome e sobrenome) de um aluno e a idade

```
#include<iostream>
using namespace std;
int main(){
    string nome;
    int idade;
    cout << "\n Inserir nome completo: ";
    cin >> nome;
    cout << "\n Inserir idade: ";
    cin >> idade;
    cout << "\n" << nome << " tem " << idade << "anos ";
    return 0;
}
```

O programa funciona quando é digitado um espaço em branco?

Leitura de dados V

- Em determinadas ocasiões, deseja-se coletar dados que contenham *strings* com tabulações, espaços em branco e/ou novas linhas
- O operador >> ignora tais caracteres
- Para englobar essas situações, C++ oferece o uso da função membro `getline`

Leitura de dados VI

```
#include<iostream>
using namespace std;
int main(){
    string nome;
    int idade;
    cout << "\n Inserir nome completo: ";
    getline(cin, nome);
    cout << "\n Inserir idade: ";
    cin >> idade;
    cout << "\n" << nome << " tem " << idade << "anos";
    return 0;
}
```

Atribuição I

Atribuir um valor de uma expressão a uma variável **significa calcular o valor** daquela expressão e **copiar** aquele valor para uma **determinada variável**.
O operador de atribuição é o sinal de igual (=)

A esquerda do operador de atribuição deve existir somente o nome de uma variável.

=

A direita, deve haver uma expressão cujo valor será calculado e armazenado na variável.

Expressão I

- Uma expressão é um conjunto de operações aritméticas, lógicas ou relacionais utilizados para fazer "cálculos" sobre os valores das variáveis.
  Ex: `a + b`
  Calcula a soma de *a* e *b*

Expressão II

- Uma constante é uma expressão e como tal, pode ser atribuída a uma variável (ou em qualquer outro lugar onde uma expressão seja necessária)
  Ex: `a = 10;`
- Uma variável também é uma expressão
  Ex: `a = b;`
- `<expressão> + <expressão>`: calcula a soma de duas expressões.
  Ex. `a = a + b;`

Operadores I

- Aritméticos: `+ - * /` e o sinal negativo: `-` ;
- Relacionais:

Operador	**Significado**	**Operador**	**Significado**
&&	e (*and*)	||	ou (*or*)
!	não (*not*)	<	menor
<=	menor igual	>	maior
>=	maior igual	==	igual
! =	diferente		

- Operadores de Ponteiros: * (acessa o conteúdo do ponteiro), & (obtém o endereço de uma variável);

Operadores II

- Precedência é a ordem na qual os operadores serão calculados quando o programa for executado. Em C, os operadores são calculados na seguinte ordem:
  - * e /, na ordem em que aparecem na expressão.
  - %
  - + e −, na ordem em que aparecem na expressão.

Operadores III

- Incrementais: ++ (incrementa) ; -- (decrementa);
  - ++a: incrementa primeiro a e depois a usa (pré-fixado)
  - a++: primeiro usa a variável e depois a incrementa (pós-fixado)
- Atribuição, combinação dos outros operadores, são eles: $=$, $+=$, $-=$, $*=$, $/=$, %=

Incremento(++) e Decremento(- -) I

- Operadores de incremento e decremento tem duas funções: servem como uma expressão e incrementam ou decrementam o valor da variável ao qual estão associados em uma unidade. Ex: c++ incrementa o valor da variável c em uma unidade
- Dependendo da posição do operador de incremento e decremento, uma função é executada antes da outra.

Incremento(++) e Decremento(- -) II

- operador à esquerda da variável: Primeiro a variável é incrementada, depois a expressão retorna o valor da expressão. Ex:

```
#include <iostream>
using namespace std;
int main () {
    int a = 10;
    cout << ++a;
    return 0;
}
```

Qual é valor de a?

Incremento(++) e Decremento(- -) III

- operador à esquerda da variável: Primeiro a variável é incrementada, depois a expressão retorna o valor da expressão. Ex:

```
#include <iostream>
using namespace std;
int main () {
    int a = 10;
    cout << ++a;
    return 0;
}
```

Imprime 11

Incremento(++) e Decremento(- -) IV

- operador a direita da variável: Primeiro a expressão retorna o valor da variável, e depois a variável é incrementada. Ex:

```
#include <iostream>
using namespace std;
int main (void) {
    int a = 10;
    cout << a++;
    return 0;
}
```

Incremento(++) e Decremento(- -) V

- operador a direita da variável: Primeiro a expressão retorna o valor da variável, e depois a variável é incrementada. Ex:

```
#include <iostream>
using namespace std;
int main (void) {
    int a = 10;
    cout  << a++;
    return 0;
}
```

Imprime 10

Incremento(++) e Decremento(- -) VI

- Em uma expressão, os operadores de incremento e decremento são sempre calculados primeiro (tem maior precedência)

```
#include <iostream>
using namespace std;
    int main (void) {
    int a = 10;
    cout << a * ++a);
    return 0;
}
```

Qual é o valor de *a*?

Incremento(++) e Decremento(- -) VII

- Em uma expressão, os operadores de incremento e decremento são sempre calculados primeiro (tem maior precedência)

```
#include <iostream>
using namespace std;
    int main (void) {
    int a = 10;
    cout << a * ++a;
    return 0;
}
```

Imprime 121

Incremento(++) e Decremento(- -) VIII

Quais são os valores impressos?

```
#include <iostream>
using namespace std;
int main () {
   int a = 10;
   cout << a++ << endl;
   cout << a << endl;
   return 0;
}
```

Atribuições simplificadas I

Uma expressão da forma

```
a = a + b;
```

onde ocorre uma atribuição a uma das variáveis da expressão pode ser simplificada como

```
a += b;
```

Atribuições simplificadas II

Comando	Exemplo	Corresponde a:
$+=$	$a+=b$	$a=a+b;$
$-=$	$a-=b$	$a=a-b;$
$*=$	$a*=b$	$a=a*b;$
$\%=$	$a\%=b$	$a=a\%b;$

Atribuições simplificadas III

Qual é o valor mostrado na tela?

```
/*Operadores*/
int main(){
    int i = 10, j = 20;
    i = i + 1;
    i++;
    j -= 5;
    cout <<"i + j = " << i+j;
    return 0;
}
```

FIM