



BCC221

Programação Orientada a Objetos

Prof. Marco Antonio M. Carvalho

2014/2



UFOP

Endereços Importantes

- Site da disciplina:
<http://www.decom.ufop.br/marco/>
- Moodle:
www.decom.ufop.br/moodle
- Lista de e-mails:
bcc221-decom@googlegroups.com
- Para solicitar acesso:
<http://groups.google.com/group/bcc221-decom>



UFOP



Avisos



Avisos

UFOP



Na aula passada

UFOP

- Programação Estruturada
- Orientação a Objetos
 - Conceitos básicos
 - Objetos
 - Classes
 - Relacionamentos entre classes
 - Análise
 - Projeto
- UML
 - Conceitos básicos
 - Diagramas de Classes
- Programação Estruturada vs. POO



Na aula de hoje

UFOP

- Processo de Criação de um Programa C++
- Programando em C++
 - Operadores
 - Palavras reservadas
 - *cin*
 - *cout*
 - Blocos de código
 - Referências
 - Ponteiros e Alocação Dinâmica
 - Sobrecarga de Funções



- C++ é uma extensão da linguagem C
 - Desenvolvida por Bjarne Stroustrup na década de 80, nos laboratórios Bell;
 - Baseada em C e Simula67.
- A maioria dos sistemas operacionais de hoje são escritos em C/C++.



"C faz com que dar um tiro no pé seja fácil; C++ torna isso mais difícil, mas quando nós o fazemos, arrebenta com a perna toda."

- Bjarne Stroustrup





- Programas em C++ consistem de peças
 - As classes e funções;
- Podemos criar cada peça nós mesmos
 - No entanto, a maioria dos programadores tiram vantagem das ricas coleções de classes e funções da **Biblioteca Padrão C++ (*C++ Standard Library*)**.
- De fato, existem duas partes a serem aprendidas
 - A linguagem C++ em si;
 - A utilização das classes e funções da biblioteca padrão C++.



- A vantagem de criar nossas próprias classes e funções é que conhceremos exatamente como funcionam;
- A desvantagem é o tempo gasto e a complexidade de projeto;
- Vamos evitar reinventar a roda
 - Utilizaremos as peças existentes sempre que possível
 - O que se chama **Reuso de Software**.



- Um dos problemas com a programação estruturada é que as unidades geradas não refletem entidades do mundo real com precisão
 - Com isso, a reusabilidade é dificultada.
- Com a orientação a objetos, se o projeto é bem feito, há uma grande tendência a reutilização.



- A linguagem C é um subconjunto da linguagem C++
 - Ou seja, C++ suporta todas as instruções e estruturas da linguagem C;
 - De fato, é possível programar em C puro e ser interpretado como C++.
- É possível criar programas híbridos de C/C++
 - Misturar programação estruturada com orientação a objetos;
 - Mas vamos deixar isso para a Maratona de Programação.

Processo de Criação de um Programa C++



Criando um Programa C++

UFOP

- Qualquer editor simples serve para criarmos nossos arquivos
 - Porém, existem editores específicos para programação, que são mais úteis.
 - No *Windows*, podemos usar o *Notepad++*
 - <http://notepad-plus-plus.org/download>
 - No *Ubuntu*, podemos usar o *Gedit*.



Criando um Programa C++

UFOP

- Editores como os exemplificados possuem atalhos padronizados:
 - *ctrl+n*: novo arquivo;
 - *ctrl+s*: salva o arquivo;
 - *F3* ou *ctrl+f*: abre a janela de busca;
 - *ctrl+r*: abre a janela de substituição;
 - *ctrl+g*: salta para determinada linha;
 - *ctrl+l*: apaga a linha inteira;
 - *ctrl+z*, *ctrl+c*, *ctrl+v*, *ctrl+x*, etc.
- *Zoom*: segure *ctrl* e use o *scroll* do mouse.



Criando um Programa C++

UFOP

- Uma vez selecionado um editor, digitamos as instruções que desejamos
 - Ou seja, o **corpo** do programa;
 - Salvamos o arquivo com a extensão **.cpp**, **.cxx**, **.cc** ou **.C** (maiúsculo);
 - O editor vai “colorir” nosso programa, ressaltando as instruções da linguagem, de forma a melhorar a organização.



Criando um Programa C++

UFOP

- Usaremos a interface de **linha de comando** para compilarmos nossos programas
 - O que significa que não utilizaremos ambientes de desenvolvimento integrado, ou IDEs
 - *Code::Blocks;*
 - *MS Visual Studio;*
 - *Etc.*
 - No Ubuntu, usamos o terminal;
 - No *Windows*, usamos o *Prompt* ou o *MSYS*, que simula o terminal do Linux, juntamente com o ***MinGW***.



MinGW

UFOP

- *Minimalist GNU for Windows*
 - <http://www.mingw.org/>
- Fornece um conjunto de programação *Open Source*;
- Grátis
 - *Tutorial sobre a instalação:*
 - <http://www.decom.ufop.br/marco/ensino/bcc221/tutorial-mingw/>



GNU

UFOP

- *GNU* é um sistema operacional tipo *Unix*, idealizado por *Richard Stallman*;
- De onde veio este nome?
 - ***GNU is Not Unix (GNU Não é Unix)***.





- O GCC (*GNU Compiler Collection*) é um conjunto de compiladores, produzido pela GNU
- Neste conjunto, temos o compilador G++
 - Para C++.





MinGW

UFOP

- Resumindo, *MinGW* é um *GNU* que roda em *Windows*
 - E tem o *G++*, que usaremos como compilador.
- Nos laboratórios também temos o sistema operacional **Ubuntu**
 - Que também possui o *G++* instalado, basta acessar o terminal.
- O uso do *G++* no *Windows* e *Ubuntu* é igual
 - O aluno escolhe qual vai usar.



Como Compilar com o G++?

UFOP

- No terminal, acesse a pasta onde está seu código-fonte, usando o comando ***cd***:
 - `cd pasta1/pasta2/pasta3`
 - A barra “/” significa que uma pasta está dentro da outra
 - Neste exemplo, `pasta3` está dentro de `pasta2` que está dentro de `pasta1`.
 - Para verificar o conteúdo de uma pasta, utilize o comando ***ls***:
 - `ls`



Como Compilar com o G++?

UFOP

- Uma vez na pasta correta, digite o seguinte comando:
 - `g++ arquivo.cpp -o programa -Wall`
 - Neste exemplo, substitua o nome “*arquivo*” pelo nome do seu arquivo.
 - Substitua também o nome *programa* pelo nome que você desejar.
 - O “-o” significa que o programa compilado se chamará *programa*;
 - O “-Wall” significa que pedimos que sejam mostrados todos os erros e avisos de compilação.



Como Compilar com o G++?

UFOP

- Se o seu programa não apresentar erros, nenhuma mensagem será apresentada pelo compilador
 - Você já pode executar seu programa.
- Caso contrário, o compilador tentará te avisar onde está o erro do seu programa
 - Será necessário corrigir o erro em seu código fonte e compilá-lo novamente.



Como Compilar com o G++?

UFOP

A screenshot of a terminal window titled "MINGW32:/c/Documents and Settings/Marco Antonio/Desktop". The window contains the following text:

```
Marco Antonio@MARCO /c/Documents and Settings/Marco Antonio/Desktop
$ g++ meu_programa.cpp -o meu_programma -Wall
Marco Antonio@MARCO /c/Documents and Settings/Marco Antonio/Desktop
$
```

The terminal has a blue header bar and a light yellow body. It features standard window controls (minimize, maximize, close) in the top right corner.



Como Compilar com o G++?

UFOP

A screenshot of a terminal window titled "MINGW32:/c/Documents and Settings/Marco Antonio/Desktop". The window contains the following text:

```
Marco Antonio@MARCO /c/Documents and Settings/Marco Antonio/Desktop
$ g++ meu_programa.cpp -o meu_programma -Wall
meu_programma.cpp: In function 'int main()':
meu_programma.cpp:7:4: error: expected ';' before 'return'
```

The terminal prompt shows the user's name, the computer name, the directory path, and the command entered. The output of the command shows a compilation error for the file "meu_programma.cpp". The error message indicates that the compiler found a syntax error in the "main()" function definition, specifically expecting a semicolon (";") before the "return" keyword at line 7, column 4.

O compilador tenta indicar em qual linha está o erro, o qual é o tipo de erro. Neste exemplo, o erro está na linha 4, e se refere à falta de ponto e vírgula.



Como Compilar com o G++?

UFOP

- Toda vez que alterarmos nossos códigos, não podemos nos esquecer de salvar antes de compilar
 - Senão, estaremos compilando uma versão antiga de nossos programas, que não refletem as alterações realizadas.



Como Executar um Programa?

UFOP

- Uma vez compilado, seu programa pode ser executado pelo próprio terminal
 - Basta digitar `./` antes do nome do programa;
 - `./programa`
- Por enquanto, nossos programas não terão janelas, *mouse* e *etc.*.
- Eles serão executados dentro do próprio terminal.



Como Executar um Programa?

UFOP

A screenshot of a terminal window titled "MINGW32:/c/Documents and Settings/Marco Antonio/Desktop". The window contains the following text:

```
Marco Antonio@MARCO /c/Documents and Settings/Marco Antonio/Desktop
$ g++ meu_programa.cpp -o meu_programa -Wall

Marco Antonio@MARCO /c/Documents and Settings/Marco Antonio/Desktop
$ ./meu_programma
Este é o meu primeiro programa na linguagem C++
Tchau!
Marco Antonio@MARCO /c/Documents and Settings/Marco Antonio/Desktop
$
```

The terminal window has a blue header bar and a light yellow body. It includes standard window controls (minimize, maximize, close) in the top right corner.



Compilando e Executando

UFOP

- No site da disciplina há diversas informações sobre compilação/execução por linha de comando e utilização da linha de comando

<http://www.decom.ufop.br/marco/ensino/bcc221/>

Programando em C++



Programando em C++

UFOP

- Veremos que parte da sintaxe de C++ é igual ou muito parecida com a sintaxe da linguagem C
 - Estruturas de fluxo são mantidas;
 - Entrada e saída simplificadas;
 - Para as novas funcionalidades, novas sintaxes.



Programando em C++

UFOP

Palavras-chave comuns a C e C++

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			



Programando em C++

UFOP

Palavras-chave exclusivas de C++

and	and_eq	asm	bitand	bitor
bool	catch	class	compl	const_cast
delete	dynamic_cast	explicit	export	false
friend	inline	mutable	namespace	new
not	not_eq	operator	or	or_eq
private	protected	public	reinterpret_cast	static_cast
template	this	throw	true	try
typeid	typename	using	virtual	wchar_t
xor	xor_eq			



Programando em C++

UFOP

- O que mais é mantido:
 - Operadores relacionais;
 - Atribuições simplificadas;
 - Incremento e decremento;
 - Operadores lógicos;
 - Ponteiros;
 - Funções;
 - Desvio Condicional
 - *if, if-else, operador ternário (?:) e aninhamentos.*
 - Estrutura de seleção
 - *switch-case.*
 - Estruturas de repetição
 - *while, do-while, for* e aninhamentos.



Programando em C++

UFOP

Operadores (precedência)	Associatividade	Tipo
::	Esquerda para direita	Definição de escopo
()	Esquerda para direita	Parênteses
++, --, static_cast<tipo>()	Esquerda para direita	Unário (pós fixo)
++, --, +, -, !	Direita para esquerda	Unário (pré fixo)
*, /, %	Esquerda para direita	Multiplicativo
+, -	Esquerda para direita	Aditivo
<<, >>	Esquerda para direita	Inserção/extração
<, <= > >=	Esquerda para direita	Relacional
==, !=	Esquerda para direita	Igualdade
&&	Esquerda para direita	E lógico
	Esquerda para direita	OU lógico
:?	Direita para esquerda	Condicional
=, +=, -=, *=, /=, %=	Direita para esquerda	Atribuição

Instrução de Saída



Programando em C++

UFOP

```
// Programa de impressão de texto.  
#include <iostream> // permite que o programa gere saída de dados na tela  
using namespace std;  
  
// a função main inicia a execução do programa  
int main()  
{  
    cout << "Welcome to C++! "<<endl; // exibe a mensagem  
  
    return 0; // indica que o programa terminou com sucesso  
} // fim da função main
```



Programando em C++

UFOP

■ Assim como em C:

- Comentários de uma linha começam com `//`;
 - `/*` e `*/` para mais que uma linha.
- Usamos a diretiva de compilação `#include` para incluir bibliotecas;
- Precisamos do `main`
 - Apesar de ser uma linguagem orientada a objetos, C++ precisa que exista a função `main`;
 - Usamos `return` o.
- `;` no final das instruções;
- Usamos `{` e `}` para delimitar um bloco de instruções.



Programando em C++

UFOP

- O que há de novo:
 - **iostream** é a biblioteca com os fluxos de *bytes* padrão (entrada/saída)
 - Não tem **.h**;
 - Equivalente à **stdio.h**.
 - **using namespace std;**
 - Um *namespace* é um grupo de nomes de entidades, como classes, objetos e funções
 - Neste caso, *std* é o agrupamento das entidades da biblioteca padrão C++;
 - Para usar os identificadores da **iostream**, precisamos indicar qual é o *namespace*.



Programando em C++

UFOP

- **endl**
 - *end line;*
 - Manipulador de fluxo;
 - Quebra a linha e limpa o *buffer*
 - Nada fica acumulado no *buffer*.



Programando em C++

UFOP

- A **entrada e saída** de dados é realizada basicamente em 3 padrões:
 - Entrada Padrão (**cin**)
 - Normalmente o teclado, mas pode ser redirecionado.
 - Saída Padrão (**cout**)
 - Normalmente o vídeo, mas pode ser redirecionado.
 - Fluxo Padrão de Erros (**cerr**)
 - Utilizado para mostrar mensagens de erro padronizadas;
 - Normalmente associado ao vídeo.
- Todos são objetos inclusos no *namespace std*.



Programando em C++

UFOP

```
cout << "Welcome to C++!\\n";
```

- É uma instrução de saída;
- << é o operador de **inserção de fluxo**
 - O valor à direita do operador é inserido no fluxo de saída.
- Similar ao *printf*, porém, simplificado.



Programando em C++

UFOP

- O objeto **cout** permite estabelecer o tamanho de um campo para impressão, através do uso de **manipuladores de tamanho de campo**:
 - **setw**
 - Define o tamanho do campo a ser impresso.
 - **setprecision**
 - Define o número de casas decimais.
 - **setfill**
 - Define o caractere usado para preencher espaços de um campo.
 - **setiosflags**
 - Define a apresentação entre ponto decimal, notação científica, etc.
- Estes manipuladores estão definidos na biblioteca **iomanip**.



Programando em C++

UFOP

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

int main()
{
    int lapis = 45, borracha =2345, caneta=420;

    cout<<"Lapis      "<<setw(12)<<lapis<<endl;
    cout<<"Borracha  "<<setw(12)<<borracha<<endl;
    cout<<"Caneta    "<<setw(12)<<caneta<<endl;

    return 0;
}
```



Programando em C++

UFOP

■ Saída

Lapis	45
Borracha	2345
Caneta	420



Programando em C++

UFOP

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

int main()
{
    int lapis = 45, borracha = 2345, caneta=420;

    cout<<setfill('.');
    cout<<"Lapis " <<setw(12)<<lapis<<endl;
    cout<<"Borracha " <<setw(12)<<borracha<<endl;
    cout<<"Caneta " <<setw(12)<<caneta<<endl;

    return 0;
}
```



Programando em C++

UFOP

■ Saída

Lapis	45
Borracha	2345
Caneta	420



Programando em C++

UFOP

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

int main()
{
    float lapis = 4.5, borracha = 2.345, caneta=4.20;

    cout<<setiosflags(ios::fixed)//ponto decimal (nao exponencial)
        <<setiosflags(ios::showpoint)//imprime o ponto decimal
        //sempre
        <<setprecision(2);//duas casas decimais
    cout<<"Lapis " <<setw(12)<<lapis<<endl;
    cout<<"Borracha " <<setw(12)<<borracha<<endl;
    cout<<"Caneta " <<setw(12)<<caneta<<endl;

    return 0;
}
```



Programando em C++

UFOP

■ Saída

Lapis	4 . 50
Borracha	2 . 35
Caneta	4 . 20



Programando em C++

UFOP

- Ainda é possível no **cout**:
 - Limitar o tamanho de campos envolvendo *strings*
 - Manipulador **setw**.
 - Manipular bases numéricas
 - Manipuladores **hex**, **oct**, **func**.
 - Imprimir caracteres gráficos
 - Tabela ASCII.

Instrução de Entrada



Programando em C++

UFOP

```
#include <iostream> // permite ao programa realizar entrada e saída
using namespace std;

// a função main inicia a execução do programa
int main()
{
    int number1; // primeiro inteiro a adicionar
    int number2; // segundo inteiro a adicionar
    int sum; // soma de number1 e number2

    cout << "Enter first integer: "; // solicita dados ao usuário
    cin >> number1; // lê primeiro inteiro inserido pelo usuário em number1

    cout << "Enter second integer: "; // solicita dados ao usuário
    cin >> number2; // lê segundo inteiro inserido pelo usuário em number2

    sum = number1 + number2; // adiciona os números; armazena o resultado em sum

    cout << "Sum is " << sum << endl; // exibe sum; termina a linha

    return 0; // indica que o programa terminou com sucesso

} // fim da função main
```



Programando em C++

UFOP

- Assim como em C:
 - O operador de atribuição é **=**;
 - Usamos os tipos primitivos;
 - Declaramos variáveis e constantes
 - Vetores e matrizes também.
 - Utilizamos os operadores aritméticos
 - Soma, subtração, multiplicação, divisão e módulo (resto);
 - Símbolos iguais.
 - Precêndia de operadores;
 - Utilização de parênteses.



Programando em C++

UFOP

- `cin >> number1;`
 - É uma instrução de entrada;
 - O objeto `cin` aguarda um fluxo de entrada;
 - `>>` é o operador de extração de fluxo
 - O valor na entrada padrão é inserido na variável à direita do operador;
 - Não utiliza formatadores.
- `cout << "Sum is " << sum << endl;`
 - Não utiliza formatadores.



Programando em C++

UFOP

- Podemos utilizar o operador de extração (>>) várias vezes na mesma instrução para lermos diversos valores
 - Este operador entende um espaço em branco como o fim de uma entrada
 - Digitamos múltiplas entradas separadas por **espaços** ou **enter**.



Programando em C++

UFOP

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

int main()
{
    float lapis, borracha, caneta;

    cin >>lapis>>borracha>>caneta;
    cout<<"Lapis " <<setw(12)<<lapis<<endl;
    cout<<"Borracha " <<setw(12)<<borracha<<endl;
    cout<<"Caneta " <<setw(12)<<caneta<<endl;

    return 0;
}
```



Programando em C++

UFOP

- Novamente, não é necessário utilizar formatadores
 - `%d`, `%c`, etc;
 - O `scanf` funciona em C++.
 - No entanto, a leitura utilizando o objeto **cin** automatiza esta tarefa.
- Vejamos o exemplo anterior, alterando o tipo das variáveis.



Programando em C++

UFOP

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

int main()
{
    int lapis, borracha, caneta;

    cin >>lapis>>borracha>>caneta;
    cout<<"Lapis " <<setw(12)<<lapis<<endl;
    cout<<"Borracha " <<setw(12)<<borracha<<endl;
    cout<<"Caneta " <<setw(12)<<caneta<<endl;

    return 0;
}
```



Programando em C++

UFOP

- Frequentemente quando lemos caracteres do teclado enfrentamos o problema de ler '\n';
- Utilizamos o método **cin.ignore** para evitar o '\n'

```
cin.ignore(256, '\n');
```

- Lê da entrada até encontrar um '\n', o qual será removido da entrada
- Não se preocupem com métodos por enquanto, veremos o assunto na próxima semana.



Programando em C++

UFOP

- Ainda é possível no `cin`:
 - Manipular bases numéricas
 - Manipuladores `hex`, `oct`, `func.`
- Veremos na próxima semana a classe `string`
 - Leitura de strings com espaços;
 - Manipulação de strings;
 - Conceitos de classe, objeto e métodos.

Blocos de Código



Programando em C++

UFOP

- Assim como em C, usamos { e } para delimitar **blocos de código**
 - No entanto, o comportamento vai além.
- Outro aspecto interessante é que variáveis podem ser declaradas em qualquer parte do código
 - Uma variável criada dentro de um bloco de código não é *visível* fora dele.



Programando em C++

UFOP

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

int main()
{
    int i=5;

    {//início do bloco
        int i = 150;
        cout<<i<<endl; //imprime 150
    }//fim do bloco

    cout<<i<<endl; //imprime 5
    return 0;
}
```



Programando em C++

UFOP

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

int main()
{
    float soma=0;

    for(int i=0; i<10; i++)
    {
        int nota;
        cin>>nota;
        soma+=nota;
    }
    //i=0 ERRO!
    cout<<soma/10<<endl;
    return 0;
}
```



UFOP



**Continua na
próxima aula...**

Referências



Programando em C++

UFOP

- O operador de **referência** & também é utilizado em C++
 - Utilizado para criar um tipo de dado chamado **referência**
 - O operador é utilizado junto ao tipo, e não à variável.
 - O termo referência é utilizado como um outro nome para uma variável já existente;
 - Uma referência não é uma cópia de uma variável
 - É a mesma variável, sob um nome diferente.
 - Toda referência deve ser obrigatoriamente inicializada.



Programando em C++

UFOP

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

int main()
{
    int n;
    int& a = n;//referência

    n=15;//também altera a

    cout<<n<<endl;
    cout<<a<<endl;

    a = 0;//também altera n

    cout<<n<<endl;
    cout<<a<<endl;

    return 0;
}
```



Programando em C++

UFOP

- Um dos usos de referências é na **passagem por referência**
 - A função pode acessar as variáveis originais da função que a chamou;
 - Note que há uma inversão em relação à linguagem C
 - Na chamada da função, passamos apenas o nome da variável;
 - No cabeçalho da função, usamos o operador de referência junto ao tipo;
 - No corpo da função não é necessário tratamento especial.



Programando em C++

UFOP

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

void troca(int& c, int& d)//referências a 'a' e 'b'
{
    int temp;

    temp = c;
    c = d;
    d = temp;
}

int main()
{
    int a = 10, b=5;

    cout<<a<<" "<<b<<endl;
    troca(a, b);
    cout<<a<<" "<<b<<endl;

    return 0;
}
```



Programando em C++

UFOP

- Note que também existem ponteiros em C++
 - Logo, é possível realizar a passagem por referência usando ponteiros;
 - No entanto, a princípio não há necessidade de fazê-lo por ponteiros.

Ponteiros e Alocação Dinâmica



Programando em C++

UFOP

- C++ possui dois operadores que substituem a finalidade das funções *malloc* e *calloc*
 - *new* e *delete*;
 - Mais adequados
 - Não retornam ponteiros void;
 - Retornam ponteiros do tipo adequado ao necessário;
 - Não é necessário utilizar *cast*.



Programando em C++

UFOP

- O operador **new** solicita memória ao sistema operacional
 - Retornando um ponteiro para o primeiro *byte* da memória alocada.
- O operador **delete** libera a memória alocada anteriormente pelo operador new
 - Devolve a memória ao sistema operacional;
 - Antes da aplicação do operador delete, a memória alocada pode ser utilizada normalmente.



Programando em C++

UFOP

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

int main()
{
    char *caractere;

    caractere = new char//alocando a memória
    *caractere = 'M';

    cout<<caractere;

    delete caractere;//liberando a memória

    return 0;
}
```



Programando em C++

UFOP

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

int main()
{
    int *numeros, tam, i;

    cout<<"Digite a quantidade de numeros"<<endl;
    cin>>tam;

    numeros = new int[tam];//alocando um vetor na memória

    for(i=0; i<tam; i++)
        numeros[i] = i+1;

    for(i=0; i<tam; i++)
        cout<<numeros[i]<<" ";

    delete [] numeros;//liberando a memória, necessário [] para vetores

    return 0;
}
```



Programando em C++

UFOP

- Para realizar a alocação de matrizes dinâmicas usando o **new**
 - Alocamos um vetor de ponteiros;
 - Em cada posição do vetor de ponteiros alocamos outro vetor
 - O agrupamento destes vetores de vetores formam uma matriz dinâmica.
- Alguns livros dizem que não é possível alocar dinamicamente uma matriz bidimensional utilizando a instrução **new**...



Programando em C++

UFOP

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

int main()
{
    int **matrix, n;
    cout << "Digite o tamanho do matriz: ";
    cin >> n;
    matrix = new int*[n];
    for (int i=0; i<n; i++)
        matrix[i] = new int[n];

    for (int l=0; l<n; l++)
        for (int c=0; c<n; c++){
            cout << "Insira um numero: ";
            cin>>matrix[l][c];
        }

    for (int i=0; i<n; i++)
        delete [] matrix[i];
    delete [] matrix;

    return 0;
}
```



Programando em C++

UFOP

- Porquê usar **new** e **delete** ao invés de **malloc** ou **calloc**?
 - Quando utilizamos objetos, temos o conceito de construtor e destrutor
 - Não é necessário *cast*;
 - Se criarmos um objeto com **new**, o construtor será chamado;
 - Se liberarmos um objeto com **delete**, o destrutor será chamado;
 - Isso não ocorre com **malloc** e **calloc**.

Sobrecarga de Funções



Programando em C++

UFOP

- Sobrecarregar funções significa criar uma família de funções que tenham o mesmo nome, mas uma lista de parâmetros diferente
 - Em número ou tipo.
- Por exemplo, vamos considerar uma função que calcula o cubo de um número
 - Que tipo de número?
 - Inteiro ou real?
- Criamos uma função para cada, e o próprio programa escolhe a mais adequada baseada no tipo do parâmetro.



Programando em C++

UFOP

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

int cubo(int num)
{
    return num*num*num;
}
float cubo(float num)
{
    return num*num*num;
}
double cubo(double num)
{
    return num*num*num;
}
int main()
{
    cout<<cubo(4)<<endl;
    cout<<cubo(12.34)<<endl;
    cout<<cubo(4567.89)<<endl;
    return 0;
}
```

Estruturas



Programando em C++

UFOP

- Saber lidar com estruturas é meio caminho para aprender a lidar com classes e objetos
 - Vimos estruturas em Introdução a Programação e AEDs I.
- Definir uma estrutura não cria uma variável
 - Define um molde para criação de variáveis.
- A linguagem C++ expande as capacidades das estruturas
 - Geralmente são utilizadas para armazenar somente dados, como em C;
 - Em C++ as estruturas podem armazenar dados e funções.



Programando em C++

UFOP

- A diferença na sintaxe de estruturas e classes é mínima
 - Basicamente, classes definem membros privados por padrão;
 - Estruturas definem membros públicos por padrão.
- A maior parte dos programadores C++ utiliza estruturas para dados e classes para dados e funções
 - A sintaxe para definição de estruturas é a mesma utilizada em C.

Arquivos



Programando em C++

UFOP

- Operações relacionadas a arquivos em C++ utilizam objetos e métodos
 - Como ainda não vimos como implementar classes e objetos, fica adiada esta parte.



Comentários Finais



Programando em C++

UFOP

- Outras coisas em que valem a pena dar uma olhada:
 - Funções inline
 - Uma evolução das macros.
 - Classes de armazenamento
 - *extern*
 - Variáveis compartilhadas entre arquivos diferentes.
 - *register*
 - Manipulação de registradores.



Programando em C++

UFOP

■ Algumas bibliotecas básicas de interesse

Biblioteca	Finalidade
iostream	Funcionalidades de entrada/saída (C++)
cmath	Funções numéricas
iomanip	Manipuladores de entrada/saída (C++)
cctype	Manipulação de caracteres
ctime	Funções de tempo
cstring	Manipulação de strings (string.h)
string	Manipulação de strings (C++)



UFOP



Perguntas?



Na próxima aula

UFOP

- Classes
- Objetos
- Métodos
 - Construtores
 - Destrutores
 - Objetos como Parâmetros de Métodos
 - Métodos que Retornam Objetos
- Separando a Interface da Implementação
- Composição: Objetos como Membros de Classes
- Funções Amigas
- Sobrecarga de Operadores
- O Ponteiro *This*



UFOP



FIM