Minicurso de C++

Aula 01
Introdução e Conceitos Básicos

Responsáveis:
Gabriel Daltro Duarte
Rafael Guerra de Pontes
Walisson da Silva Soares
Weslley da Cunha Santos
Ianes Grécia

Introdução

- Para que programar?
- Tipos de linguagem:
  - Linguagem de máquina
    - Linguagem natural de um computador
    - Consistem em *strings* de números (1s e 0s)
  - Linguagens assembly
    - Abreviações simbolizam operações elementares.
    - *Assemblers* convertem para ling. de máquina
  - Linguagens de alto nível.
    - Ex.: C/C++, Java, Pascal.

História do C++

- Desenvolvido por *Bjarne Stroustrup* no início da década de 1980 na *Bell Laboratories*.
- Deriva do C, oferecendo recursos extras.
- Dá suporte à programação orientada a objetos (POO).
- C/C++ muito usados para desenvolver Sistemas Operacionais.

Bjarne Stroustrup.
Fonte: http://www.app2us.com/images/Bjarne.jpg

Conceitos importantes

- *Código-Fonte* O texto de um programa que um usuário pode ler, normalmente interpretado como o programa. O código-fonte é a entrada para o compilador C.
- *Código-Objeto* Tradução do código-fonte de um programa em código de máquina que o computador pode ler e executar diretamente. O código-objeto é a entrada para o linkeditor.
- *Linkeditor* Um programa que une funções compiladas separadamente em um programa. Ele combina as funções da biblioteca C padrões com o código que você escreveu. A saída do linkeditor é um programa executável.
- *Biblioteca* O arquivo contendo as funções padrão que seu programa pode usar. Essas funções incluem todas as operações de E/S como também outras rotinas úteis.
- *Tempo de compilação* Os eventos que ocorrem enquanto o seu programa está sendo compilado. Uma ocorrência comum em tempo de compilação é um erro de sintaxe.
- *Tempo de execução* Os eventos que ocorrem enquanto o seu programa é executado.

Estrutura básica de um programa

```
1 #include <iostream>
2
3 int main(){
4
5     std::cout << "Bom dia, pessoas!" << std::endl;
6
7     return 0;
8
9 }
```

Comentários

- Servem para documentar seu código.
- Ajudam no entendimento do programa.
- Não influenciam nas instruções executadas.
- Tipos de comentário em C++:
  - /* (...) */ ⇒ Comentário multilinha.
  - // (...) ⇒ Comentário de linha única.

Programa simples comentado

```cpp
/*
    PET Engenharia Elétrica IFPB
    Minicurso de C++ 2015.2
    Programa de impressão de texto na tela
*/

#include <iostream> // Permite saída de dados na tela

// A função main inicia a execução do programa
int main(){

    std::cout << "Bom dia, pessoas!" << std::endl; // Exibe a mensagem

    return 0; // Indica ao SO que o programa terminou corretamente

} // Fim da função main
```

Variáveis

- Região da memória nomeada pelo programador.
- O identificador da variável
  - pode conter letras, números e ‘_’.
  - pode iniciar com qualquer letra ou ‘_’.
  - não pode iniciar com números.
  - não pode ser uma palavra reservada.
  - deve ter alguma relação com seu propósito.
  - deve ser precedido pelo tipo da variável na sua declaração.

Palavras-chave de C++

Palavras-chave comuns às linguagens de programação 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`			

Palavras-chave somente de C++

`asm`	`bool`	`catch`	`class`	`const_cast`
`delete`	`dynamic_cast`	`explicit`	`false`	`friend`
`inline`	`mutable`	`namespace`	`new`	`operator`
`private`	`protected`	`public`	`reinterpret_cast`	
`static_cast`	`template`	`this`	`throw`	`true`
`try`	`typeid`	`typename`	`using`	`virtual`
`wchar_t`				

Tipos primitivos do C++

Type	Keyword
Boolean	bool
Character	char
Integer	int
Floating point	float
Double floating point	double
Valueless	void
Wide character	wchar_t

Type	Typical Bit Width	Typical Range
char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	Range	0 to 65,535
signed short int	Range	-32768 to 32767
long int	4bytes	-2,147,483,648 to 2,147,483,647
signed long int	4bytes	same as long int
unsigned long int	4bytes	0 to 4,294,967,295
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

Exemplos de declaração de variáveis

```
1int numero;
2const float mediaAprovacao = 7.0;
3double velocidade = 40.35;
4char nome[7] = "Rafael";
5char nome2[7] = {'R', 'a', 'f', 'a', 'e', 'l', '\0'};
6string nome3;
7string nome4 = "Programar é muito divertido!";
8int identificadores[20];
9float notas[3] = { 9.5, 9.2, 10.0 };
```

Exemplo de uso de variável

```cpp
#include <iostream>

int main(){

    int x = 10;

    std::cout << "O valor de x é " << x << std::endl;

    return 0;

}
```

```
rafael@rafael-inspiron-7520:~/Desktop/MinicursoCPP/Aula01$ ./ex01
O valor de x é 10
rafael@rafael-inspiron-7520:~/Desktop/MinicursoCPP/Aula01$
```

```cpp
// Exemplo do site http://www.tutorialspoint.com/cplusplus/cpp_data_types.htm
#include <iostream>
using namespace std;

int main() {

   cout << "Size of char : " << sizeof(char) << endl;
   cout << "Size of int : " << sizeof(int) << endl;
   cout << "Size of short int : " << sizeof(short int) << endl;
   cout << "Size of long int : " << sizeof(long int) << endl;
   cout << "Size of float : " << sizeof(float) << endl;
   cout << "Size of double : " << sizeof(double) << endl;
   cout << "Size of wchar_t : " << sizeof(wchar_t) << endl;

   return 0;

}
```

```
rafael@rafael-inspiron-7520:~/Desktop/MinicursoCPP/Aula01$ ./exsizeof
Size of char : 1
Size of int : 4
Size of short int : 2
Size of long int : 8
Size of float : 4
Size of double : 8
Size of wchar_t : 4
rafael@rafael-inspiron-7520:~/Desktop/MinicursoCPP/Aula01$
```

Escopo de uma variável

- É a região do código em que uma variável pode ser acessada e/ou modificada.
- Variável global vs. Variável local.
- A seguir, exemplo que demonstra o escopo de uma variável.

```cpp
#include <iostream>
using namespace std;
int x = 1;
void funcao1(int x);
void funcao2();
int main(){
    int x = 2;
    cout << "x no início da main antes das funcoes vale " << x << endl;
    funcao1(4);
    funcao2();
    cout << "x no início da main depois das funcoes vale " << x << endl;
    {
        int x = 3;
        cout << "x dentro das chaves vale " << x << endl;
    }
    for(int x = 0; x < 3; x++){
        cout << "x dentro do for vale " << x << endl;
    }
    cout << "x depois do for dentro da main vale " << x << endl;
}
void funcao1(int x){
    cout << "x dentro da funcao1 vale " << x << endl;
}
void funcao2(){
    cout << "x dentro da funcao2 vale " << x << endl;
}
```

```
x no início da main antes das funcoes vale 2
x dentro da funcao1 vale 4
x dentro da funcao2 vale 1
x no início da main depois das funcoes vale 2
x dentro das chaves vale 3
x dentro do for vale 0
x dentro do for vale 1
x dentro do for vale 2
x depois do for dentro da main vale 2
```

Operadores e Expressões

- Operadores.
- Propriedades dos operadores.
- Tipos de Expressões: Aritméticas, Relacionais e Lógicas.
- Conversões de tipos.

Operadores

Operador	Descrição
==	Igualdade
>	Maior
<	Menor
<=	Menor igual
>=	Maior igual
!=	Diferente

Símbolo	Significados
-	Subtração
+	Adição
*	Multiplicação
/	Divisão
%	Resto da divisão (módulo)

Operador	Descrição
&&	AND
||	OR
!	NOT (operador de negação)

Tabela – Operadores lógicos

Operadores bit a bit

Operador	Ação
&	AND Lógico
|	OR Lógico
^	XOR (OR exclusivo)
~	NOT
>>	Shift Right
<<	Shift Left

Propriedades dos operadores

- Aridade: Quantidade de operandos em que o operador atua (unário, binário e ternário).
- Precedência: Determinação da ordem relativa com o que os operadores são aplicados.
- Associatividade: Determina a ordem de aplicação de operadores de mesma precedência.

Precedência de operadores

Operadores
() [] ->
! ~ ++ -- * & (tipo) sizeof()
* / %
+ -
>> <<
< <= >= >
== !=
&
^
|
&&
||
? () : ()
= += -= *= etc
,

Funções de Entrada e Saída

- Bibliotecas:
  - <iostream>
  - <iomanip>
- Objetos úteis:
  - cout ⇒ conectado à tela, por padrão.
  - cerr ⇒ conectado à tela, por padrão.
  - cin ⇒ conectado ao teclado, por padrão.
- getline() ⇒ função que lê uma linha completa.

Exemplo de E/S

```cpp
#include <iostream>

using namespace std;

int main(){

    int a, b;

    cout << "Digite um inteiro: ";
    cin >> a;
    cout << "Digite um inteiro: ";
    cin >> b;

    cout << a << " + " << b << " = ";
    cout << (a+b) << endl;

    return 0;

}
```

Exercício

Copiem, compilem e executem o seguinte código:

```cpp
#include <iostream>

using namespace std;

int main(){

        string nome;
        int numero;

        cout << "Digite seu primeiro nome: ";
        cin >> nome;
        cout << "Digite seu número da sorte: ";
        cin >> numero;

        cout << "\nBom conhecê-lo, " << nome;
        cout << "!! Então você gosta do número " << numero << "? Show!\n";

        return 0;

}
```

Estruturas de Controle

- Normalmente instruções são executadas uma após outra na ordem que estão escritas (Execução sequencial);
- O C++ possui **estruturas de controle** que permitem ao programador especificar que a próxima instrução a executar pode ser diferente da próxima instrução da sequência.

Estruturas de Controle

Foi comprovado que qualquer programa pode ser escrito utilizando apenas três tipos de estruturas de controle:

1 - Estrutura de sequência,
2 - Estrutura de seleção,
3 - Estrutura de repetição;

Estruturas de Controle

1 - Estrutura de sequência: A menos que instruído de outra maneira, o computador executa as instruções C++ uma depois da outra na ordem em que elas são escritas, isto é, em sequência.

Estruturas de Controle

2 - Estrutura de seleção: O C++ fornece três tipos de estruturas de seleção:

2.1 - Instrução de seleção if: Realiza uma ação se uma condição for verdadeira, ou pula a ação se a condição for falsa.

2.2 - Instrução de seleção if...else: Realiza uma ação se uma condição for verdadeira e realiza outra ação e a condição for falsa.

Estruturas de Controle

2.3 - Instrução de seleção switch: É uma instrução de seleção múltipla, uma vez que seleciona entre muitas ações diferentes;

2.4 - Instrução de seleção ternária : Funciona de um modo semelhante à operação if...else, apenas menos verboso.

Estruturas de Controle

Instrução de seleção if

Considere um programa que faz a leitura da nota de um aluno e imprime o texto “passou” caso sua nota seja maior que 7;

pseudocódigo:

declaração da variável nota;
leitura da variável nota;
Se nota for maior ou igual a 7
    imprima passou

Estruturas de Controle

Instrução de seleção if

A instrução Se pode ser escrita em C++ como:

```
#include <iostream>

using namespace std;

int main()
{
      float nota;
      cin >> nota;
      if(nota >= 7)
            cout << "passou" << endl;
      return 0;
}
```

Estruturas de Controle

Instrução de dupla seleção if...else

O if...else permite que o programador especifique uma ação a ser realizada caso a condição seja verdadeira e uma outra ação caso a condição seja falsa;

Considere um programa que faz a leitura da nota de um aluno e imprime o texto “passou” caso sua nota seja maior que 7 e imprime “não passou” caso sua nota seja menor que 7;

pseudocódigo:

declaração da variável nota;
leitura da variável nota;
Se nota for maior ou igual a 7
imprima “passou”
Caso contrário
imprima “não passou”;

Estruturas de Controle

Instrução de dupla seleção if...else

A instrução Se… Caso contrário pode ser escrita em C++ como:

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

int main()
{
      float nota;
      cin >> nota;
      if (nota >= 7)
            cout << "passou" << endl;
      else
            cout << "nao passou" << endl;
      return 0;
}
```

Estruturas de Controle

Instruções de seleção if...else aninhadas

As intruções if...else aninhados testam múltiplos casos fazendo combinações de estruturas if...else.

pseudocódigo:

declaração da variável nota;
leitura da variável nota;
Se nota for maior ou igual a 9
    imprima “A”
Caso contrário Se for maior ou igual a 8
    imprima “B”
Caso contrário Se for maior ou igual a 7
    imprima “C”
Caso contrário
    imprima “D”

Estruturas de Controle

Estrutura de seleção: Exercícios

1. Faça um programa em C++ que leia um número e determine se ele é par ou ímpar.

2. Faça um programa em C++ que leia a altura e o sexo (M ou F) de uma pessoa, calcule e mostre o seu peso ideal, utilizando uma das seguintes fórmulas:

Para homens: (72.7 * altura) – 58.0

Para mulheres: (62.1 * altura) – 44.7

Estruturas de Controle

Estrutura de repetição *while*

A instrução de repetição (também chamada de loop) while permite ao programador especificar que um programa deve repetir uma ação ou um conjunto de ações enquanto uma determinada condição é verdadeira.

Estruturas de Controle

Estrutura de repetição *while* - exemplo 1

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

int main()
{
      int  par = 0;
      cout  <<  “Os numeros pares de 0 a 100  sao: ” ;
      while (par <=100)
      {
            cout << par << endl;
            par = par + 2;
      }
      return 0;
}
```

Estruturas de Controle

Estrutura de repetição *while* - exemplo 2

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

int main()
{
      int nr = 1;
      while (nr != 0 )
      {
            cout << “Digite um numero (0 para finalizar):  “;
            cin >> nr;
            if (nr > 0)
                  cout << “Este numero eh positivo\n”;
            else
                cout << “Este numero eh negativo\n”;
      }
      return 0;
}
```

Estruturas de Controle

Exercício

Estrutura de repetição *while*

Escreva um programa que calcule a soma dos inteiros de 1 a 10. Use while para realizar o loop. O loop deverá terminar quando x chegar a 11.

Estruturas de Controle

Estrutura de repetição *for*:

Essa estrutura de repetição especifica os detalhes da repetição controlada por um contador em uma única linha de código.

Estruturas de Controle

Estrutura de repetição *for* - Exemplo 01

```cpp
#include <iostream>

using namespace std;

int main(){

   cout << "Os números pares entre 0 e 100 são: " << endl;
   for (int par = 0; par <= 100; par += 2){
      cout << par << " ";
   }

   cout << endl;

   return 0;
}
```

Estruturas de Controle

Estrutura de repetição *for*:

Exercício 01

Escreva um programa que leia, no máximo, 10 números não negativos e exiba a soma deles. Caso seja digitado um número negativo, deve-se interromper a leitura dos números e exibir a soma deles (mesmo que não tenham sido lidos os 10 números).

Estruturas de Controle

Estrutura de repetição *for*:

Exercício 02

Elabore um programa para calcular a potência de um número inteiro utilizando um *for*. O expoente lido também deve ser um número inteiro.

Arrays

- Definição;
- Declaração;
- Exemplos de utilização;
- Arrays multidimensionais.

Definição de Array

- Um array pode ser entendido com uma coleção de variáveis do mesmo tipo.
- Guarda uma quantidade fixa de elementos sequenciais.
- Todos os arrays são constituídos de posições adjacentes de memória.

Declarando arrays

Definição geral:

tipo nome_array[tamanho do array];

tipo ⇒ Tipo do elemento contido no array.

nome_array ⇒ Identificador do array.

tamanho do array ⇒ Inteiro positivo.

Inicialização de arrays

- Você pode inicializar os elementos de um vetor um a um ou utilizando um único comando, como:

```
float numeros[3] = {1,2,2.5};
```

- Se você omitir o tamanho do array, um array grande o suficiente para armazenar o vetor será criado.

Acessando elementos do array

Os elementos presentes num array são acessados pelo identificador do array seguido de um índice, começando pelo índice 0.

Ex:

```
float num = numeros[1];
```

Exemplo de utilização

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

int main()
{
    int vetor[10];
    for(int i =0;i<10;i++)
        vetor[i] = i*10;
    cout << "Indice" << setw(13) << "Valor" << endl;
    for(int j=0;j<10;j++)
        cout << setw(6) << j << setw(13) << vetor[j] << endl;
    return 0;
}
```

Arrays Multidimensionais

- Podemos generalizar a declaração de um array para n-dimensões referenciando o tamanho de cada dimensão.

  Ex:

```
double valores[2][3][4];
```

Arrays bidimensionais

- A forma mais simples de um array multidimensional é o array bidimensional.
- Um array bidimensional é, em essência, uma lista de arrays unidimensionais.
- Exemplo de declaração de um array bidimensional:
  - `tipo identificador[x][y];`
- Arrays bidimensionais podem ser entedidos como matrizes com x linhas e y colunas

Exercício 01

Leia 10 números inteiros e os imprima na tela em ordem inversa à que foi lida.

Exercício 02

Leia uma matriz quadrada 3x3 e exiba a sua transposta.

Bons estudos e até a próxima aula! :)