Revisão sobre C++

Cálculo Numérico

Ivo Calado

Instituto Federal de Educação, Ciência e Tecnologia de Alagoas

23 de Fevereiro de 2016

Roteiro

Estrutura básica de um programa

```
//arquivo.cpp
 #include <stdio.h> Provenientes de diretórios default
 #include "a.h" //headers entre "" são relativos ao
     arquivo atual

 int main(int arcg, char* argv[]) {
  int a, b;
  for(int i = 10; i < 20; i++) {
    int j;
  }
  return 0;
 }
```

Compilação

g++ a.cpp -o programa

Definição de variáveis

```
int main(int arcg, char* argv[]) {
        int a, b = 3, d = 4;
        return 0;
}
```

Denominação	Número de Bytes	Conjunto de valores
char	1	caracteres codificados no código ASCII
int	2	números inteiros de –32768 a 32767
long ou long int	4	números inteiros de –65536 a 65535
float	4	números reais de $-3{,}4\text{x}10^{38}$ a $-3{,}4\text{x}10^{-38}$ e $3{,}4\text{x}10^{-38}$ a $3{,}4\text{x}10^{38}$
double	8	números reais de $-1{,}7\text{x}10^{308}$ a $-1{,}7\text{x}10^{-308}$ e $1{,}7\text{x}10^{-308}$ a $1{,}7\text{x}10^{308}$
void	0	conjunto vazio

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Determinando o tamanho de uma variável

```
  /* usando o operador sizeof */

#include <stdio.h>

int main()
  {
    int nr;

    printf("A variável nr é um inteiro e tem %d bytes.\
       n",sizeof(nr));
    printf("Já o tipo de dado char tem %d bytes.\n",
       sizeof(char));

    return(0);
  }
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Constantes

Objetivo

Definir um identificador que irá possuir o mesmo valor durante toda a execução do programa

```
#define minhaConstante 20
int main(int arcg, char* argv[]) {
        int k = minhaConstante;
        const int j; //erro de compilação
        const int i = 10;
        i = 20;//erro de compilaçao
        return 0;
}
```

Expressões aritméticas

- Encontram-se definidas as operações aritméticas +, -, *, /, %
- Operações entre parênteses são executadas primeiro, seguidas de *,/ e % e, por fim, + e -
- Operadores adicionais: i++, ++i

```
int main(int arcg, char* argv[]) {
        int a = 10 + 20 * 30 - 3 % 3;
        return 0;
}
```

Expressões lógicas e comparadores

- Operadores lógicos: &&, || e !
- Operadores comparativos: <, <=, >, >=, !=, ==
- Operadores lógicos bit-a-bit: ~, &, | e ^
- Deslocamento de bits: >> e <<
- Atribuição: =, +=, -=, /=, *= %=, >>=, <<=, | =, &=

Entrada e saída de dados (estilo C)

```
 #include<stdio.h>
int main(int arcg, char* argv[]) {
        int i;
        printf("Digite um valor: ");
        scanf("%d", &i);
        printf("O valor digitado é %d\n", i);
        return 0;
 }
```

Entrada e saída de dados (estilo C++)

```
#include<iostream>

using namespace std;
int main(int arcg, char* argv[]) {
        int i;
        cout<<"Digite um valor: ";
        cin>>i;
        cout<<"O valor digitado foi: "<<i<<endl;
        return 0;
 }
```

O comando *if*

```
#include <stdio.h>
int main(int argc, char* argv[])
{
        float a, b, Maior;
        printf("Digite os dois numeros: ");
        scanf("%f %f", &a, &b);
        Maior = a;
        if (b > a)
                Maior = b;
        printf("O maior dos numeros %.2f , %.2f e' %.2f
            \n", a, b, Maior);
}
```

O comando *if-else*

```
#include <stdio.h>
int main(int argc, char * argv[]) {
        int x, y;
        printf("Digite o numero: ");
        scanf("%d", &x);
        if (x % 2 == 0)
                printf("%d e' par \n", x);
        else
                printf("%d e' impar \n", x);
}
```

O operador ternário

```
#include <stdio.h>
int main(int argc, char* argv[])
{
        float a, b, Maior;
        printf("Digite os dois numeros: ");
        scanf("%f %f", &a, &b);
        Maior = (b > a)? b : a;
        printf("O maior dos numeros %.2f , %.2f e' %.2f
            \n", a, b, Maior);
}
```

Switch

```
#include <stdio.h>
int main(int argc, char** argv) {
        int mes, numDias;
        printf("Digite o mes \n"); scanf("%d", &mes);
        switch (mes)
        {
                case 2: numDias = 29;
                break;
                case 4: case 6: case 9:
                case 11:
                        numDias = 30;
                break;
                default : numDias = 31;
        }
        printf("O mes de numero %d tem %d dias \n", mes
            , numDias);
}
```

O comando *for*

```
#include <stdio.h>
int main(int a, char** b) {
        for (int i = 1; i <= 10; i++)
                printf("%d ", i);
        return 0;
}
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

O comando *while*

```
#include <stdio.h>
int main(int a, char** b) {
        int i = 10;
        while(i--)
                printf("%d ", i);
        return 0;
}
```

O comando *do-while*

```
#include <stdio.h>
int main(int a, char** b) {
        int i = 10;
        do
                printf("%d ", i);
        while(i--);
        return 0;
}
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Vetores

- Um vetor é um conjunto de variáveis de um mesmo tipo de dado as quais são acessadas e referenciadas através da aposição de índices ao identificador do vetor

```
int main(int a, char** b) {
        int v[10];
        v[0] = 10; //Modifica a primeira posição
        v[5] = 30; //Modifica a sexta posição
        return 0;
}
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Cadeias de caracteres

- Em C não existe um tipo primitivo capaz de armazenar uma cadeia de caracteres. Para tal, faz uso de vetores de caracteres

```
int main(int a, char** b) {
        char s[10];
        printf("Digite um nome: ");
        scanf("%s", s);
        printf("Nome digitado: %s\n", s);
        return 0;
}
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Matrizes

- Matrizes são estruturas indexadas utilizadas para representar certa quantidade de valores de um mesmo tipo
- Um vetor é um tipo especial de matriz, chamado de matriz *unidimensional*
- Matrizes podem ter 1, 2, ..., n dimensões

```
int main(int a, char** b) {
        int m[10][10];

        //Acessando a linha 0 e a coluna 1
        m[0][1] = 10;
        return 0;
}
```

Definição de uma função

```
#include <stdio.h>
int soma(int a, int b) {
        return a + b;
}

int main(int a, char** b) {
        int c = soma(10, 50);
        return 0;
}
```

Função recursiva

```
#include <stdio.h>
int fat(int a) {
        if(a == 0)
                return 1;
        else
                return a * fat(a - 1);
}

int main(int a, char** b) {
        int f = fat(5);
        printf("Fatorial de 5 = %d\n", f);
        return 0;
}
```

Funções de biblioteca

- Utiliza-se a diretiva include para incluir funções e métodos definidos em outros locais
- É possível definir o seu próprio conjunto de funções, métodos e classes

Ponteiros

```
#include <stdio.h>

int main(int a, char** b) {
        int f = 10;
        int *p = &f;
        printf("f = %d; *p = %d, p = %p\n", f, *p, p);
        return 0;
}
```

f = 10; *p = 10, p = 0x7fff56d17044

Implementando a função swap

```
#include <stdio.h>

void swap(int *a, int *b) {
        int k = *a;
        *a = *b;
        *b = k;
}

int main(int a, char** b) {
        int p1 = 10, p2 = 20;
        printf("p1 = %d; p2 = %d\n", p1, p2);

        swap(&p1, &p2);
        printf("p1 = %d; p2 = %d\n", p1, p2);
        return 0;
}
```