Inovação Tecnológica e Empreendedorismo

Lógica e Arduino

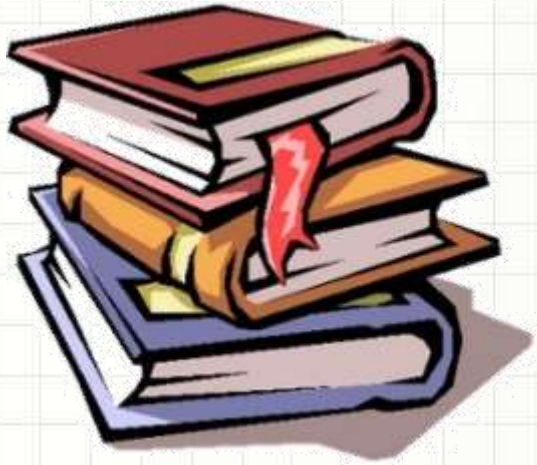
Prof. Dr. Daniel Caetano

2017 - 2

Objetivos

- Rever conceitos vetores e funções
- Compreender o uso desses conceitos por meio de aplicações no Arduino

Material de Estudo



Material	Acesso ao Material
Apresentação	http://www.caetano.eng.br/ (Inovação e Empreendedorismo – Aula C)
Material Didático	da disciplina Algoritmos
Biblioteca Virtual	“algoritmos”, “programação”

CONTEXTUALIZAÇÃO

O que são Algoritmos

- Algoritmo: fabricar vinho para venda
  - Plantar a uva
  - Colher a uva
  - Amassar a uva
  - Deixar fermentar
  - Engarrafar
  - Distribuir para a venda
- Envolve
  - Tarefas/Processos
  - Decisões

Momento Lúdico:

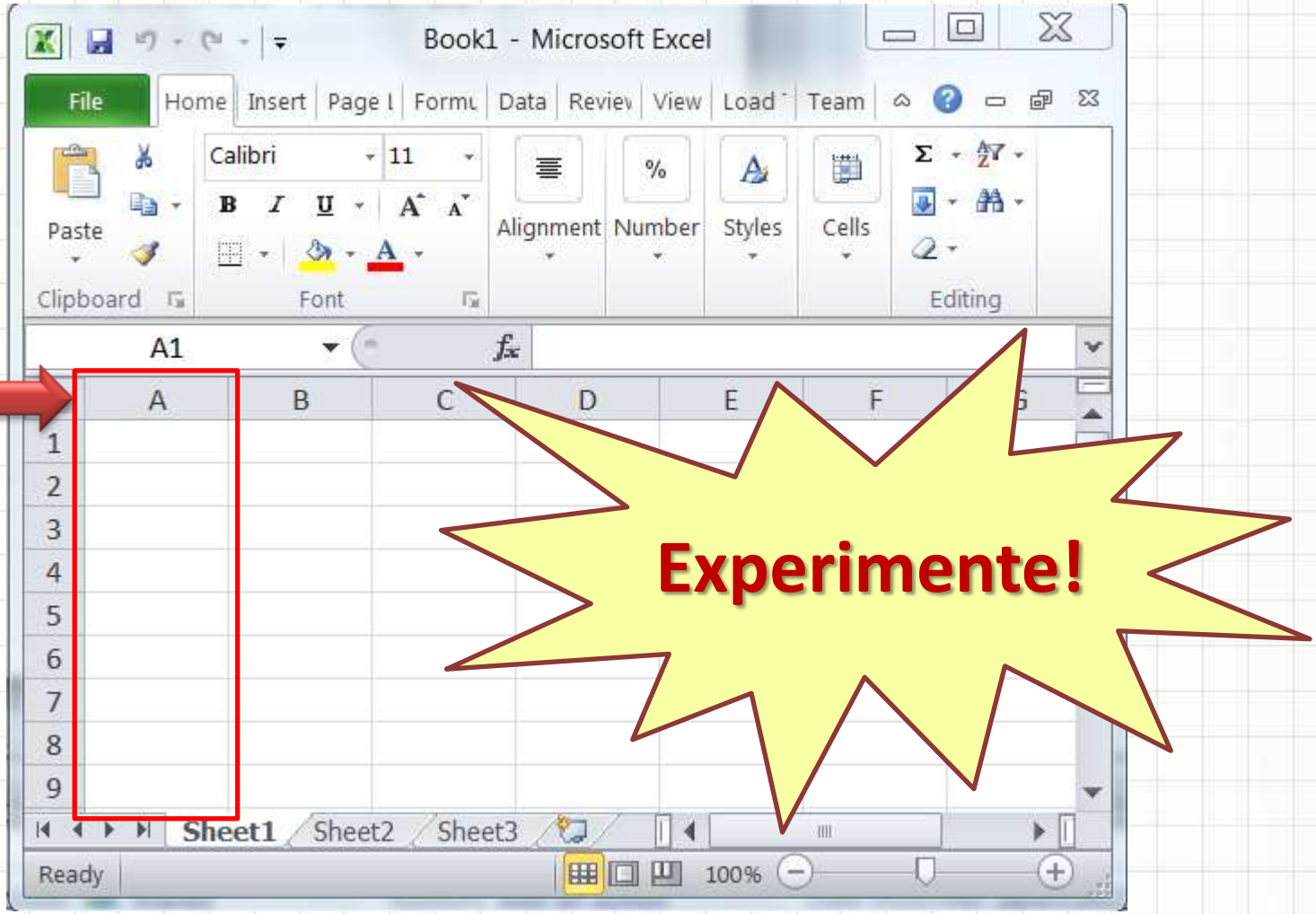
O Uso de Tabelas

Momento Lúdico

- Em que programa usamos “tabelas”?

Momento Lúdico

- Em que programa usamos “tabelas”?



Vetores e Matrizes

Vetores

- Curso de Algoritmos:
  - Quantos valores se guardava em uma variável?
- Quantos valores posso guardar aqui?

int i;

- E nessa variável aqui?

float nota;

Vetores

- Mas e se quiséssemos guardar as notas de todos os alunos da tuma (10 alunos)?

float nota1;

float nota2;

float nota3;

(...)

float nota10;

Vetores

- Vamos ver uma forma diferente de declarar!
- Quantos valores acham que cabe aqui?

float nota[10];

- Muito mais simples, não?
- Isso é um **vetor**, uma espécie de **tabela**:

nota

0	1	2	3	4	5	6	7	8	9

Vetores

- Guardando valores...

float nota[10];

nota

0	1	2	3	4	5	6	7	8	9

- Nota[0] = 10;

Vetores

- Guardando valores...

float nota[10];

nota

0	1	2	3	4	5	6	7	8	9
10									

- Nota[0] = 10;

Vetores

- Guardando valores...

float nota[10];

nota

0	1	2	3	4	5	6	7	8	9
10									

- Nota[0] = 10;
- Nota[5] = 7;

Vetores

- Guardando valores...

float nota[10];

nota

0	1	2	3	4	5	6	7	8	9
10					7				

- Nota[0] = 10;
- Nota[5] = 7;

Vetores

- Declarando um vetor preenchido

int idades[10] = { 10, 2, 30, 55, 6, 28, 32, 9, 8, 15 };

idades

0	1	2	3	4	5	6	7	8	9
10	2	30	55	6	28	32	9	8	15

Muito prático!

Matrizes

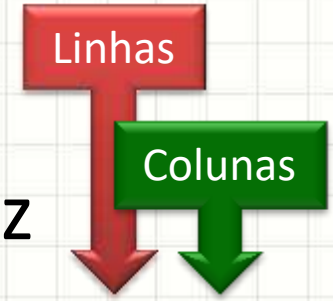
- Mais parecido com o Excel:
  - Linhas e colunas

	0	1	2
0			
1			
2			
3			

- Como criar uma matriz?

Matrizes

- Criando uma Matriz




```
int m[4][3];
```

	0	1	2
0			
1			
2			
3			

- Escrever na matriz?
- M[2][1] = 7;

Matrizes

- Criando uma Matriz

int m[4][3];

	0	1	2
0			
1			
2		7	
3			

- Escrever na matriz?
- M[2][1] = 7;

Matrizes

- Criando uma Matriz preenchida

int m[4][3] = { { 1, 2, 3}, {5, 6, 7}, {0, 1, 2}, {2, 3, 4}};

	0	1	2
0	1	2	3
1	5	6	7
2	0	1	2
3	2	3	4

- Ler da matriz?
- cout << M[3][2];
  - Imprime... 4!

Funções Simples

Momento Lúdico

- Como fazer um misto quente?

- Como fazer um sanduiche com um “recheio” genérico?

Funções Simples

- Queremos uma função que imprima a assinatura de nosso e-mail:

  Atenciosamente,

  Prof. Daniel Caetano

  prof@caetano.eng.br

```
cout << “Atenciosamente” << endl;
cout << “Prof. Daniel Caetano” << endl;
cout << “prof@caetano.eng.br” << endl;
```

  - Mas toda função precisa de um nome...!

Nome e Delimitação da Função

- Chamemos a função de “assina”:

```
assina()
{
    cout << “Atenciosamente” << endl;
    cout << “Prof. Daniel Caetano” << endl;
    cout << “prof@caetano.eng.br” << endl;
}
```

Os parênteses são importantes!

- Assim como uma receita de bolo...
  - Uma função precisa ser usada para ter resultado!
  - Vamos inserir a função em um programa!

Uso da Função

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

assina()
{
    cout << “Atenciosamente,” << endl;
    cout << “Prof. Daniel Caetano” << endl;
    cout << “prof@caetano.eng.br” << endl;
}

main()
{
    assina();
}
```

Erro?

Funções têm retorno!

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

void assina()
{
    cout << “Atenciosamente,” << endl;
    cout << “Prof. Daniel Caetano” << endl;
    cout << “prof@caetano.eng.br” << endl;
}

main()
{
    assina();
}
```

void : indica função sem retorno!

main não precisa ter retorno?!?

Funções com Retorno

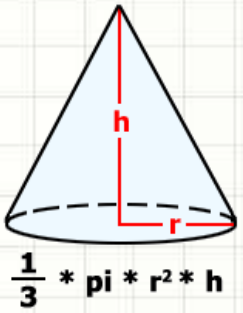
Funções com Retorno

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

float pi()
{
    return 3.14159;
}

main()
{
    cout << pi();
}
```

E se mudar o **float** por **int**?

Funções com Parâmetros

Funções com Parâmetros



- Calcular o volume de um cone

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

float volumeCone(float r, float h)
{
    return (1.0/3.0)*3.14*r*r*h;
}

main()
{
    cout << volumeCone(10,2);
}
```

O que ocorre se mudarmos esses valores?

Escopo de Variáveis e Variáveis Globais

Escopo de Variáveis

- Variáveis das funções: **locais**
  - Valem apenas dentro da função

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

float volumeCone(float r, float h) {
    float x;
    x = (1.0/3.0)*3.14*r*r*h;
    return x;
}
main() {
    volumeCone(10,2);
    cout << x;
}
```

ERRO!

Escopo de Variáveis

- Variáveis **globais**
  - Valem no programa todo

Cuidado!

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

void volumeCone(float r, float h) {
    x = (1.0/3.0)*3.14*r*r*h;
}
main() {
    volumeCone(10,2);
    cout << x;
}
```

Aplicando ao Arduino

Vetores no Arduino

- Arduino UNO / Nano R3:
  - 32KB de Flash RAM para o código (~30KB úteis)
  - 2KB de RAM para variáveis
- Inteiro (int): 4 bytes
  - Memória lota com 512 variáveis inteiras
  - int vetor[512];

```
int vetor[512];
```

Lotou a RAM de variáveis!

- Como resolver?

Vetores no Arduino

- Se os dados não forem variáveis
  - Trecho de áudio
  - Imagem
- Armazenar na RAM de programa...
  - Precisam ser globais
  - Precisam ser constantes
  - Usar a *keyword* PROGMEM

```
#include <avr/pgmspace.h>
const PROGMEM int vetor[512] = { 0, 1, ..., 511 };
```

Vetores no Arduino

- Lembrando que textos são strings...

```
#include <avr/pgmspace.h>
const PROGMEM char msg[] = { “Uma mensagem texto” };
```

- Quando se imprime na serial...
  - Também se usa RAM de variáveis!

```
Serial.print(“Uma mensagem qualquer”);
```

- Mas é possível usar a RAM de programa:

```
Serial.print(F(“Uma mensagem qualquer”));
```

Exemplo de Animação

animacao.ino

```
#include <avr/pgmspace.h>
const PROGMEM byte animacao[6][4] = {
    { 1, 0, 0, 0 }, { 0, 1, 0, 0 }, { 0, 0, 1, 0 },
    { 0, 0, 0, 1 }, { 0, 0, 1, 0 }, { 0, 1, 0, 0 },
};
byte frame = 0;
void setup() {
   pinMode(2, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
}
void loop() {
   digitalWrite(2,animacao[frame][0]);
   digitalWrite(3,animacao[frame][1]);
   digitalWrite(4,animacao[frame][2]);
   digitalWrite(5,animacao[frame][3]);
   delay(250);
   frame = frame + 1;
   if (frame > 5) frame = 0;
}
```

Exemplo de Animação

animacao.ino

```
#include <avr/pgmspace.h>
const PROGMEM byte animacao[6][4] = {
    { 1, 0, 0, 0 }, { 0, 1, 0, 0 }, { 0, 0, 1, 0 },
    { 0, 0, 0, 1 }, { 0, 0, 1, 0 }, { 0, 1, 0, 0 },
};
byte frame = 0;
void setup() {
   pinMode(2, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
}
void loop() {
   mostraFrame(frame);
   delay(250);
   frame = frame + 1;
   if (frame > 5) frame = 0;
}
void mostraFrame(byte f) {
   digitalWrite(2,animacao[f][0]);
   digitalWrite(3,animacao[f][1]);
   digitalWrite(4,animacao[f][2]);
   digitalWrite(5,animacao[f][3]);
}
```

Exemplo de Animação

animacao.ino

```
#include <avr/pgmspace.h>
const PROGMEM byte animacao[6][4] = {
    { 1, 0, 0, 0 }, { 0, 1, 0, 0 }, { 0, 0, 1, 0 },
    { 0, 0, 0, 1 }, { 0, 0, 1, 0 }, { 0, 1, 0, 0 },
};
byte frame = 0;
void setup() {
   pinMode(2, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
}
void loop() {
   mostraFrame(frame);
   delay(250);
   frame = frame + 1;
   if (frame > 5) frame = 0;
}
void mostraFrame(byte f) {
   for (byte i=0; i<=3; i=i+1) digitalWrite(i+2,animacao[f][i]);
}
```

Exemplo de Animação

Desafio

animacao.ino

```
#include <avr/pgmspace.h>
const PROGMEM byte animacao[6] = { B0001, B0010, B0100, B1000, B0100, B0010 };
byte frame = 0;
void setup() {
   pinMode(2, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(5, OUTPUT);
}
void loop() {
   mostraFrame(frame);
   delay(250);
   frame = frame + 1;
   if (frame > 5) frame = 0;
}
void mostraFrame(byte f) {
    for (byte i=0; i<=3; i=i+1)
        digitalWrite(i+2,(animacao[f]>>i)&1);
}
```

PERGUNTAS?

Conclusões

Resumo

- Vetores e Matrizes
  - Prático para armazenar muitos dados
- Funções
  - Útil para organizar o programa
- Aplicações no Arduino
  - Armazenamento na Memória Flash
  - Economizar RAM

- Apresentação?
- O que é o conhecimento?

Parte Prática

Parte Prática

- Por que realizar uma parte prática?

Trabalho para Hoje

- Debata com as vantagens e desvantagens das alternativas propostas na aula anterior.
- Selecione a alternativa mais adequada e responda às questões A e B da lista no site do professor, referentes à Aula 03.
- Comece a trabalhar na implementação, usando o simulador ou o Arduino.