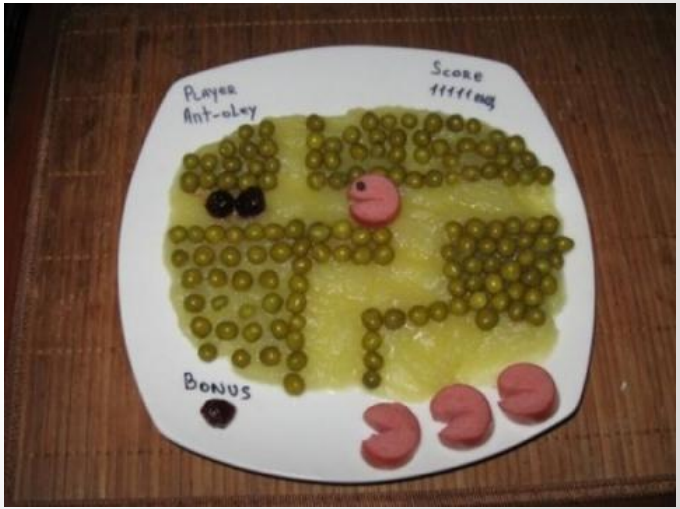
Registros e Enumerações

Aula 21

O que é um registro?

- Registros (**structs**) são estruturas compostas que nos permitem guardar vários valores heterogêneos
  - Mistureba!



Serve pra….

- Criarmos nossos próprios tipos de variáveis!

```
struct aluno{
    char nome[100];
    int XP;
    int habilidades[20];
}
```

Oh Yeah, Rock on!

- Para definirmos um registro, precisamos da seguinte estrutura:

```
struct Nome_Registro{
   campos do registro
};
```

Campos de um registro

- Os campos podem ser qualquer tipo de estrutura que a gente viu até agora
  - Tipos de variáveis primitivos (char, int, float, double…)
  - Tipos de variáveis compostas (vetores, matriz, string)
  - Ponteiros
  - Outros Registros!

Exemplo 1

```
struct filme{
   char titulo[20];
   int ano;
   int oscars;
   int assistiu;
};
```

E para declarar no programa?

```
struct nome_registro nome_variável
```

- Ou logo após a declaração do registro, se quiser!

```
struct nome_registro{
   campos
} nome_variável1, nome_variável2...;
```

Exemplo 1

```
struct aluno{
   char nome[50];
   int XP;
   int habilidade[20];
};

struct aluno al1;
```

Exemplo 2

```
struct turma{
   char nome[20];
   struct aluno alunos[60];
} imd0012;
```

Ou seja mais chique!

- Você pode declarar oficialmente um tipo novo de variável!

```
typedef struct nome_registro{
   campos
} Nome_Tipo;

Nome_Tipo nome_variavel;
```

Exemplo

```
typedef struct aluno{
   char nome[50];
   int XP;
   int habilidade[20];
} Aluno;

Aluno aluno1;
```

E pra inicializar?

- Pode fazer que nem vetor!

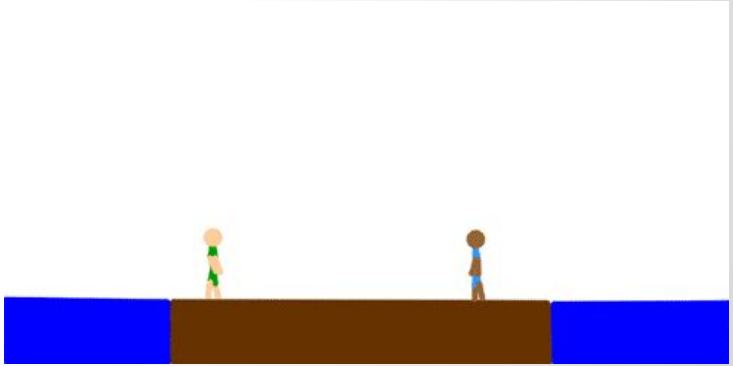
```
struct aluno aluno1 = {“Mari do XP”, 1000, {100, 200, 50, 150, 80}}

imd0012 = {“ITP”, {“Chianc”, 700, {100, 200, 80, 100}}};
```

E pra manipular os campos?

- Para acessar os campos, basta colocar o operador . da seguinte fora:
  - **variavel.campo**

```
aluno.XP = 100;
printf(“%s”, aluno.nome);
```



Tem ponteiro pra Registro?

- Teeeeeem! Igual a qualquer variável!
  - Ponteiro terá o tamanho da soma dos campos do registro
  - Use **sizeof** para pegar o tamanho em bytes de um registro

```
Aluno a;
Aluno *snitch;
snitch = &a;
```

Operador especial

- Para pegar um valor de um campo de um registro através de um ponteiro teríamos que fazer
  - `(*registro).campo`

- Mas existe um operador especial para a gente não ter que escrever assim!
  - `registro->campo`

Exemplos com ponteiros

```
Aluno a;
Aluno *b;
b = &a;

printf(“%s”, b->nome);
b->XP += 100;
```

Brain Crazy Time

```
typedef struct {
   int *ptr1;
   int *ptr2;
} Registro;
```

```
int i1 = 100;
Registro reg, *reg_ptr;
reg.ptr1  = &i1;
reg_ptr   = &reg;
*reg.ptr1 = 35;
*(*reg_ptr).ptr1 = *reg.ptr1 / 7;
*reg_ptr->ptr1   = *reg_ptr->ptr1 + 5;

printf( "%i, %i, %i, %i\n", i1, *reg.ptr1,
*(*reg_ptr).ptr1, *reg_ptr->ptr1 );
```

E pode passar pra função?

- Poooooooode!
- Funciona igual a variáveis **(copia o valor)**
- Pode ser passado por referência **com um ponteiro (mais eficiente)**

Exemplos com funções

```
typedef struct {
    int hora;
    int minuto;
    int segundo;
} Horario;

Horario time = {10, 40, 0}
segundos = converte_horas(time)
```

Exemplos com funções

```
int converte_horas(Horario h){
    return h.hora*3600 + h.minuto*60 + h.segundo;
}
```

E se passasse **por referência?**

```
int converte_horas(Horario *h){
    return h->hora*3600 + h->minuto*60 + h->segundo;
}
```

Exemplos com funções

Pode retornar um registro também!

```
Horario despertador(Horario h){
   h.hora+=1;
   h.minuto+=30;
   return h;
}
```

Constantes

- Podemos usar constantes para deixar partes do nosso código mais legíveis
  - Ex: Em um jogo de damas, poderíamos ter:

```
#define PRETO 0
#define BRANCO 1

if(peca.cor == PRETO)
```

Mas fica ruim com muitas...

```
#define PEAO 0
#define TORRE 1
#define CAVALO 2
#define BISPO 3
#define RAINHA 4
#define REI 5
```

- E pra lembrar quem é quem depois?

Enumerações

- Cria tipos especiais de dados, que associam constantes com valores numéricos (0...N)

```
enum pecas {
     peao,
     torre,
     bispo,
     cavalo,
     rei,
     rainha
};
```

Podem ser definidas como tipos

- Pode ser criado como um tipo de dado, assim como os structs

```
typedef enum dias{
     domingo,
    segunda,
     terca,
     quarta,
     quinta,
    sexta,
    sábado
} Dia; //Nome do tipo de dado
```

No código...

```
int main(){
   Dia hoje = terca;
   Dia amanha = (hoje+1)%sabado;

   if(hoje == terca || hoje == quinta)
      printf(“Tem aula de ITP!\n”);
```

- Fica melhor de ler!

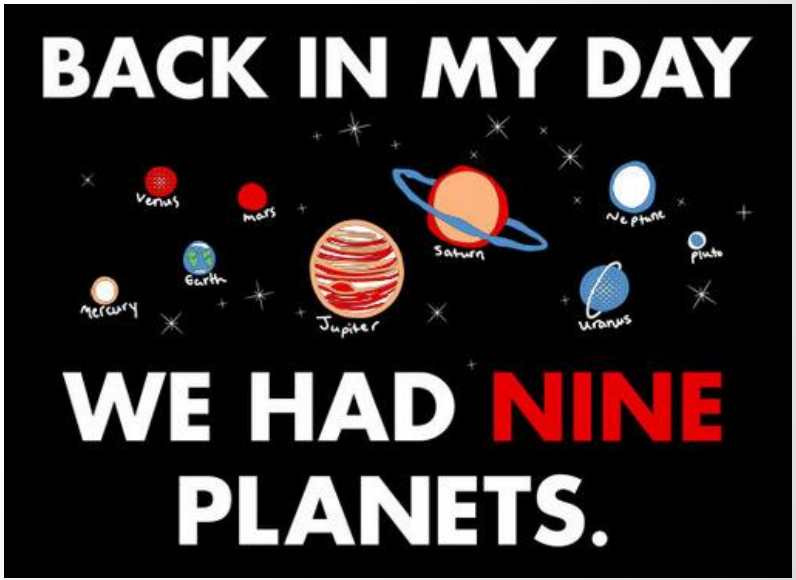
Cuidados..

- Não é um string!!!!
  - strcmp(“rainha”, pecas)
- Se colocar para imprimir, sairá o valor numérico associado à enumeração

Pode-se alterar o valor padrão!

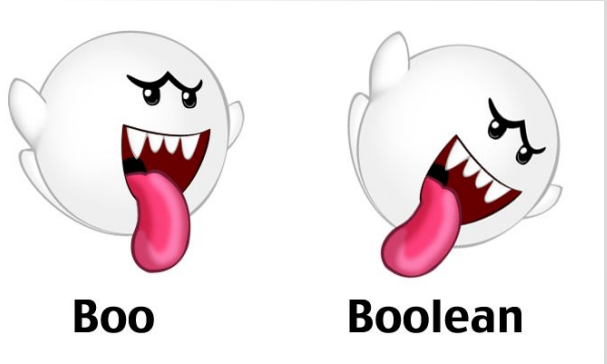
- Não precisa ser necessariamente de 0 a N-1…

```
typedef enum planetas{
      mercurio = 0,
      venus = 10,
      terra = 20,
      marte = 30,
      jupiter = 40,
      saturno = 50,
      uranio = 60,
      netuno = 70,
      plutao = 80
}
```



Exemplos

- Dá pra criar o tipo de dado que falta pra gente né?




```
typedef enum bool{
     false,
     true
} Boolean;

int main(){
     Boolean ok; //Uma variável do tipo booleano! A multidão delira!
}
```

Dúvidas?

Problema 01

- Poker 2 - a revanche!
  - No poker existe as seguintes possibilidades de mão:
    - **Carta mais alta:** o maior valor em mãos (Ace High, Às vale mais)
    - **Par** - duas cartas de mesmo valor
    - **Par duplo** - Dois pares
    - **Trinca** - três cartas de mesmo valor
    - **Straight** - sequência crescente de cartas
    - **Flush** - cinco cartas do mesmo naipe
    - **Full House** - um par e uma trinca
    - **Quadra** - quatro cartas de mesmo valor
    - **Straight Flush** - sequência crescente do mesmo naipe
    - **Royal Straight Flush** - sequência crescente do naipe de diamante

James McHold’em

- Seu programa irá ler cinco cartas da entrada
  - Cada carta será descrita pelo seu valor (2 - 10, J, Q, K e A), e pelo seu naipe (C - copas, D - diamante, E - espadas e P - paus)

- Seu programa deve determinar qual a jogada na mão do jogador

Exemplo de entrada

AC 10P 5P 3D KE	Carta Alta
4P 4C 5E 6D QC	Par
QC QD QP QE KE	Quadra
5C 6C 7D 8E 9P	Sequencia

Problema 02

Imprima um calendário usando enumerações para associar os dias da semana.

ex: dia 1 é quinta feira, logo:

DOM SEG TER QUA QUI SEX SAB

				1	2	3
4	5	6	7	8	9	10
11	12	13	14	.....