Aula 18 - Registros

Lord Vader

Roteiro da Aula

Objetivos: conhecer as estruturas heterogêneas de armazenamento de informações.

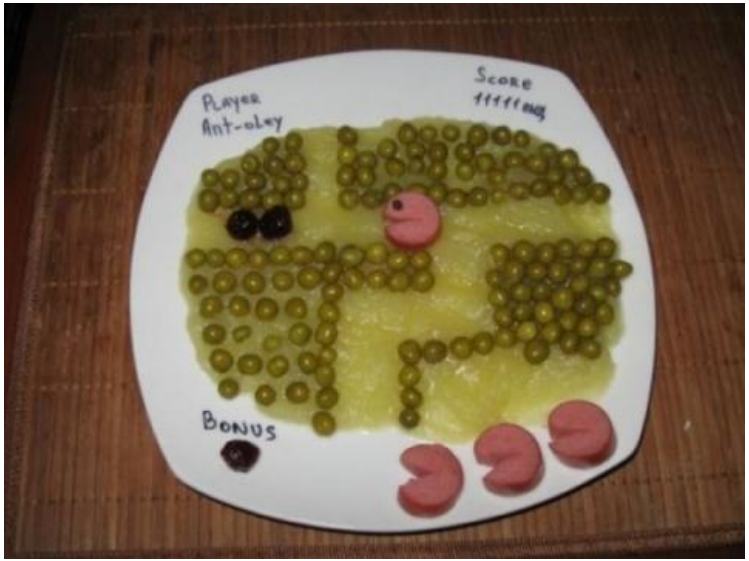
Fluxo:

- Registros
- Definição de tipos
- Enumerações

Do jeito que você quiser!

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 habilidade[20];
};

struct aluno al1;
```

Exemplo 2

```
struct turma{
   char nome[20];
   struct aluno alunos[40];
} 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 = {“Joaquim sextou!”, {100,
200, 50, 150, 80}}


imd1012 = {“ITP”, {“Vai dormir Mathias!”, {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?

- Tem sim senhor! 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

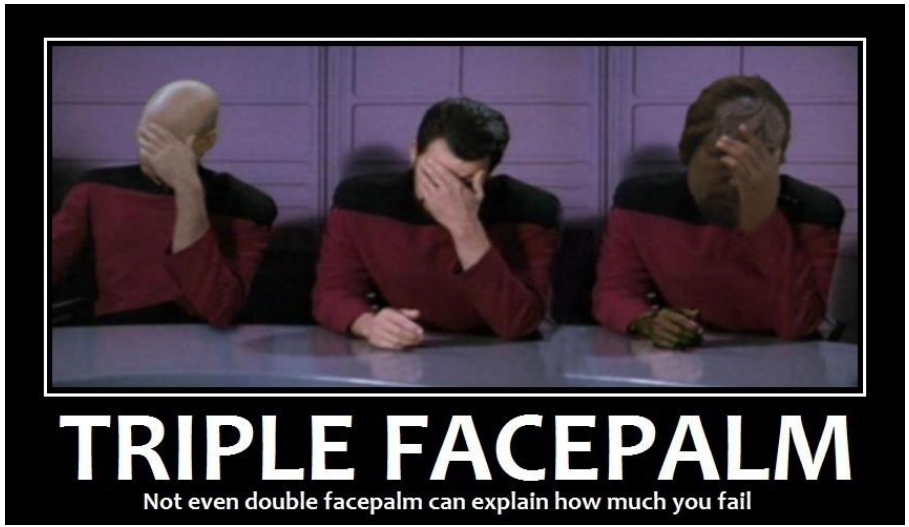
```
typedef struct {
   int *ptr1;
   int *ptr2;
} Registro;
```

```
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!

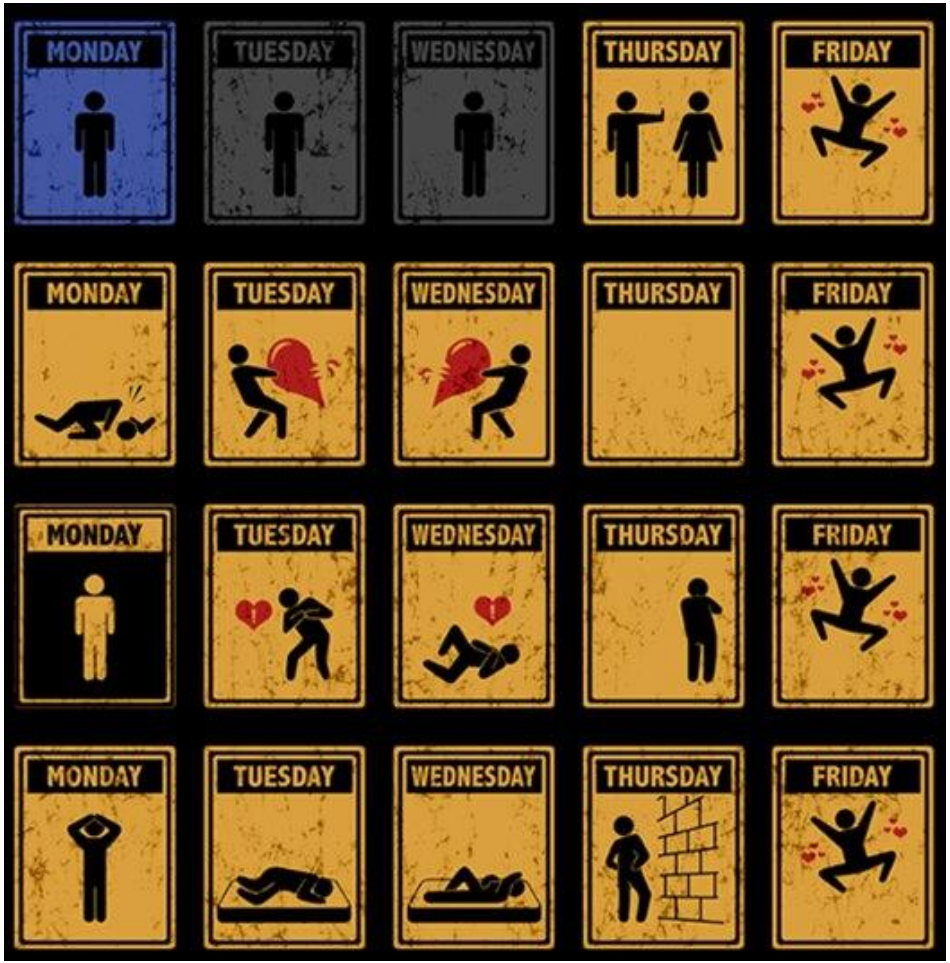
```
Horario despertador(Horario h){
   h.hora+=1;
   h.minuto+=30;
   return h;
}
```

Intervalo da cabeça

Bulindo com Registros

Vejam o arquivo com nome IMDiary para o exercício de hoje (não é uma atividade, é apenas para exercitar :))

Dúvidas?


MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY