

struct

- O `struct` permite definir um novo tipo de dados, formado pelo agrupamento de vários dados de tipos já existentes.
- Cada um dos dados que fazem parte da `struct` é denominado membro da `struct`.
- O acesso a um membro da `struct` é feito utilizando o nome da `struct` e o nome do membro, separados por um ponto: `nome_struct.nome_membro`
- Ao passar como parâmetro para uma função ou retornar de uma função um objeto do tipo `struct`, são passados/recebidos todos os membros da `struct`.
- As `structs` também podem ser passadas por cópia, endereço ou referência.
 - `structs` grandes geralmente são passadas por referência, para evitar a cópia desnecessária de muitos bytes, mesmo que não se pretenda alterá-las na função.
 - Quando não se pretende alterar o parâmetro, deve-se indicá-lo como `const`.

SEM STRUCT	COM STRUCT
<pre>void imprime(int *V_x, int V_N) { for (int i=0; i<V_N; i++) cout << V_x[i] << ' '; } int main(void) { int *A_elem; int A_dim; ... imprime(A_elem, A_dim); }</pre>	<pre>struct { int *x; int N; } Vetor; void imprime(const Vetor &V) { for (int i=0; i<V.N; i++) cout << V.x[i] << ' '; } int main(void) { Vetor A; ... imprime(A); }</pre>

typedef

- O `typedef` permite criar um novo nome (um apelido) para um tipo de dados já existente.
 - `typedef tipo_existente novo_nome`
 - É utilizado para facilitar a programação e compreensão do programa.
 - O tipo de dados pode ser referenciado pelo novo nome ou pelo nome já existente anteriormente, indiferentemente.

```
typedef int* ptr_int;
int main(void)
{
    int i = 13;
    ptr_int P = &i; // P é um ponteiro para inteiro que aponta para o endereço de i
    *P = 7;
    cout << i;      // Imprime 7
}
```