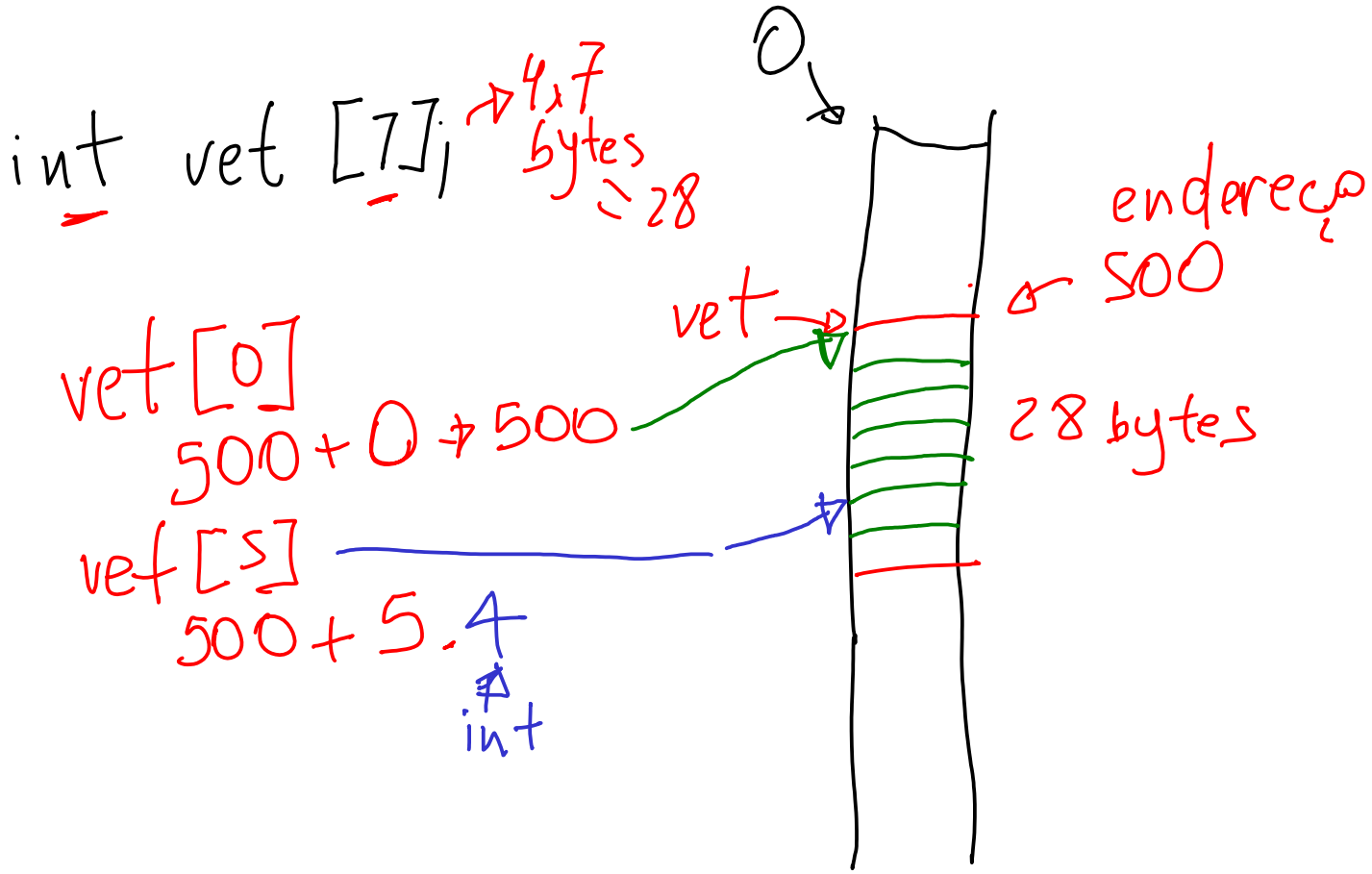
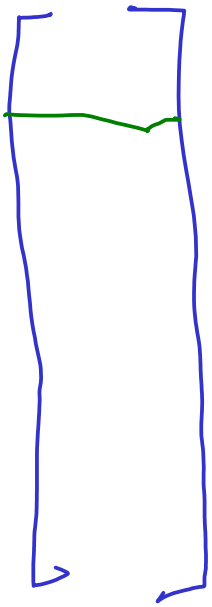

int vet [7];
4x7 bytes = 28
endereço
500
vet
28 bytes
vet [0]
500 + 0 → 500
vet [5]
500 + 5.4
≠ int

```
struct Aluno {
  10-char nome[10];
   4-int idade;
   8-float ira;
  40-float notas[5];
};
Aluno A;
```

e=100

100,A



A.nome[4] → 100 + 4 · 1 → 104

A.notas[7] ⊘ Permitido

122 + 7.8 = 178

tam Aluno?

sizeof(Aluno) → 62 bytes

```
void swap( int *a , int *b) {
    int x = a[0];
    a[0] = b[0];
    b[0] = x;
}
```

```
int vet[10];
```

end: `vet + 5`

saltos: `vet[5]`

`int *vet`

`vet[5] == *(vet + 5)`

igual

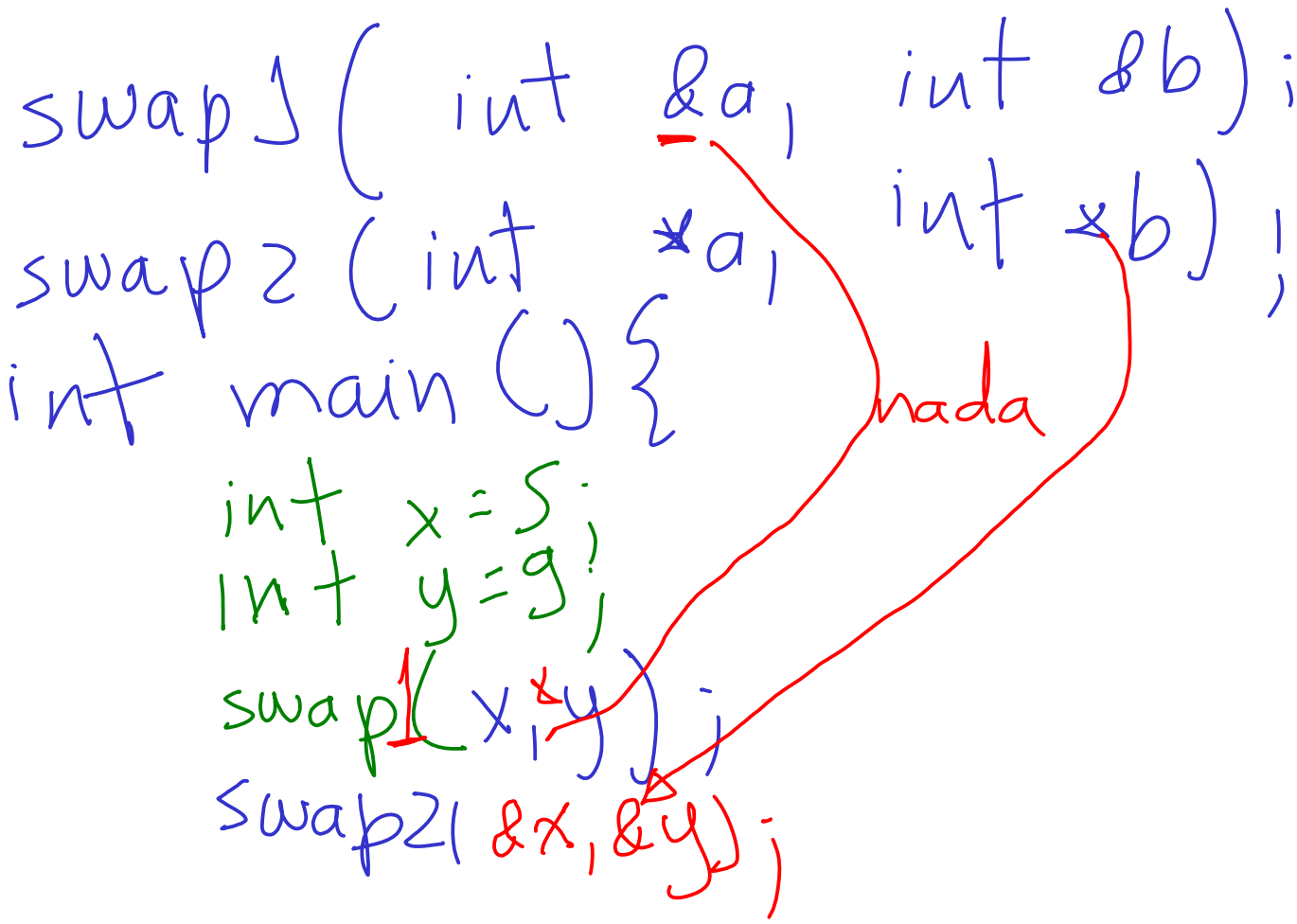
```
void swap (int *a, int *b){
    int x = *(a+0);
    *a = *b
    *b = x;
}
```

res é vetor → [soma, sub, mult, div]

```
void calc (int *x, int *y, float *res){
    res[0] = x[0] + y[0];
    res[1] = *(x+0) - *y;
    *(res+2) = *x * *y;
    res+=3;
    *(res) = (*x)/(*y)
}
```

```
void calc(int *x, int *y, float *res);
int main() {
    int x[1] = {4};
    int y[1] = {7};

    float res[4];
    calc(&x, &y, res)
}
```


swap1(int &a, int &b);
swap2 (int *a, int *b);
int main (){
int x=5;
int y=9;
swap1(x, y);
swap2(&x, &y);
nada

int *p; ← não é um vetor!

int vet[]; → não é um vetor!

* p → ponteiro

vet[] → ponteiro estático

&a → ponteiro fiel.

int vet[5];

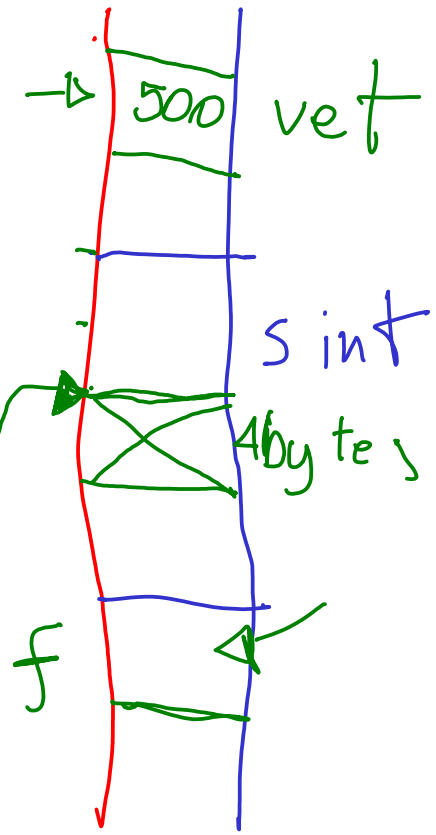
cout << vet;

float * f;

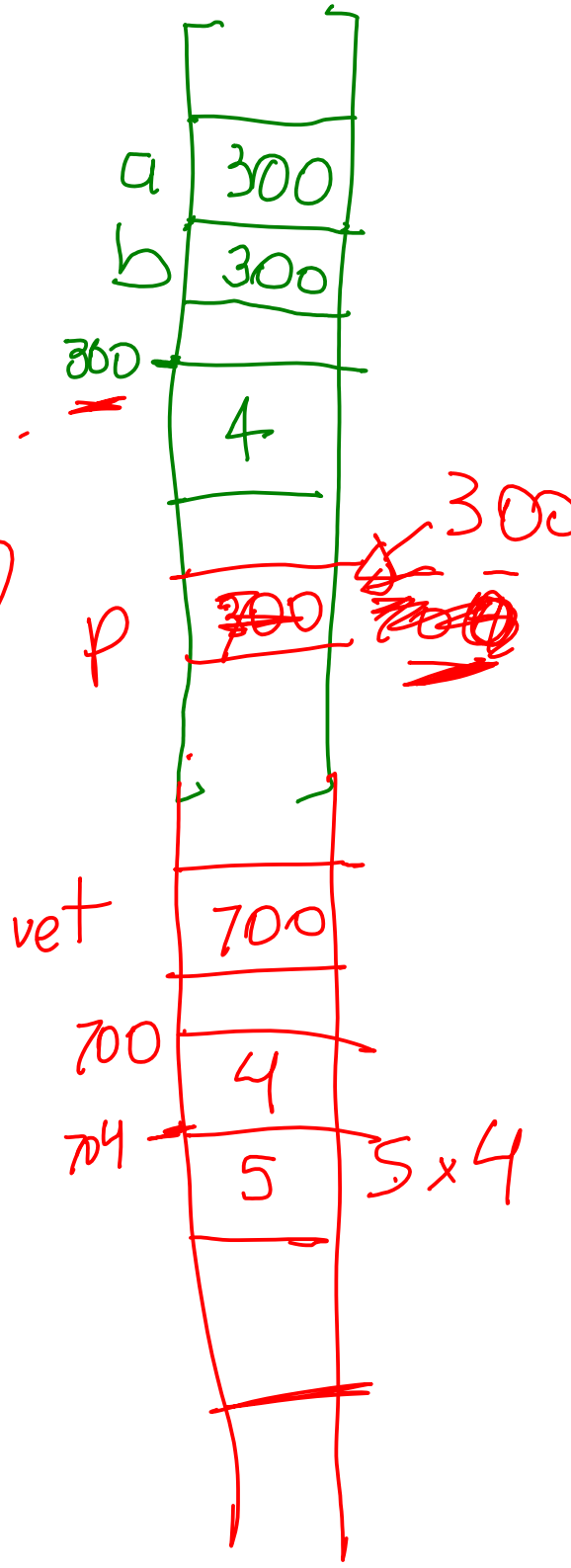
vet[2]

500+2.4

508



```
int a = 4;
int &b = a;
b = 7;
int *p = &a;
int vet[5];
p = vet;
p++;
*p = 5;
p--;
*p = a;
```



P = &a

cópia

```
int soma (int a, int b);
```

referência

```
void swap (int &a, int &b);
```

endereço

```
void swap (int *a; int *b);
```