Aula 13:
- Estruturas (parte 2)

Prof. Jesús P. Mena-Chalco
jesus.mena@ufabc.edu.br

3Q-2017

Estruturas (=registros)

Estruturas

```c
#include<stdio.h>

int main()
{
    struct ponto3D {
        double x;
        double y;
        double z;
    };

    struct ponto3D p1;   /*um registro p1 do tipo ponto3D*/

    printf("%ld\n", sizeof(p1));
    printf("%f %f %f\n", p1.x, p1.y, p1.z);

}
```

```
24
0.000000 0.000000 0.000000
```

maiorDistancia.c

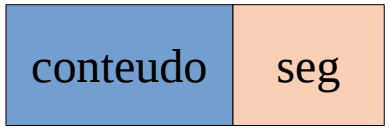
```c
#include <stdio.h>
#include <math.h>

struct ponto3D {
    double x;
    double y;
    double z;
};

double dEuclidiana(struct ponto3D p, struct ponto3D q) {
    return sqrt(pow(p.x-q.x,2) + pow(p.y-q.y,2) + pow(p.z-q.z,2));
}


int main() {
    int i, j, n;
    double maiorD = 0;

    scanf("%d", &n);
    struct ponto3D v[n] ;

    // Leitura dos pontos
    for (i=0; i<n; i++)
        scanf("%lf %lf %lf", &(v[i].x), &(v[i].y), &(v[i].z) );

    // Busca pela maior distancia entre quaisquer 2 pontos 3D
    for (i=0; i<n-1; i++)
        for (j=i+1; j<n; j++)
            if ( maiorD < dEuclidiana(v[i], v[j]) )
                maiorD = dEuclidiana(v[i], v[j]);

    printf("%lf", maiorD);
}
```

maiorDistancia.c

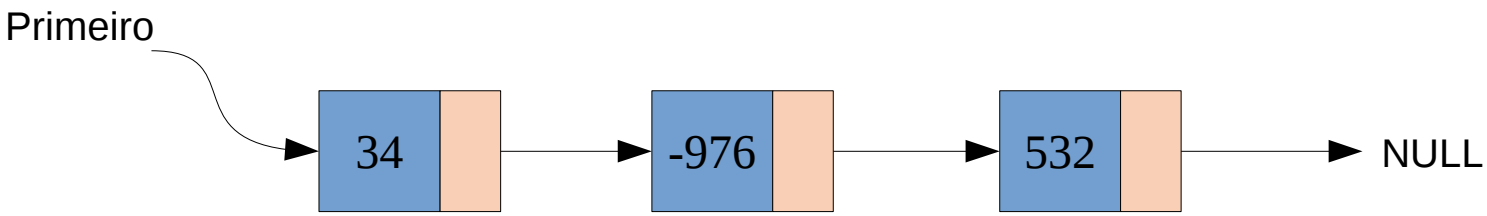
```
$ gcc maiorDistancia.c -lm -o maiorDistancia.exe

$ ./maiorDistancia.exe < vetorDePontos3D.txt

$ 173.205081
```

Lista ligada (versão simples)

Uma lista ligada




```c
struct cel {
    int conteudo;
    struct cel *seg;
};

typedef struct cel celula;
```



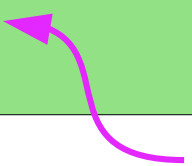
listaLigada1.c

```c
int main() {
    int i, n, numero;

    scanf("%d", &n);
    celula* primeiro = NULL;

    // Leitura dos pontos e criacao dos elementos
    for (i=0; i<n; i++) {
        scanf("%d", &numero);
        primeiro = inserirElemento(primeiro, numero);
    }
}
```

```c
celula* inserirElemento(celula* primeiro, long int numero) {
    celula* novo   = (celula *) malloc(sizeof(celula));
    if (novo==NULL){
        printf("\nNao foi possivel criar celula");
        exit(1);
    }
    novo->seg      = NULL;
    novo->conteudo = numero;

    celula* p = primeiro;
    if (p==NULL)
        primeiro = novo;
    else {
        while (p->seg!=NULL)
            p = p->seg;
        p->seg = novo;
    }
    return primeiro;
}
```

listaLigada1.c

```
$ gcc listaLigada1.c -o listaLigada1.exe

$ ./listaLigada1.exe < vetor101.txt
$ ./listaLigada1.exe < vetor102.txt
$ ./listaLigada1.exe < vetor100k.txt
```



Porque seu desempenho é ruim?

Exercício 2

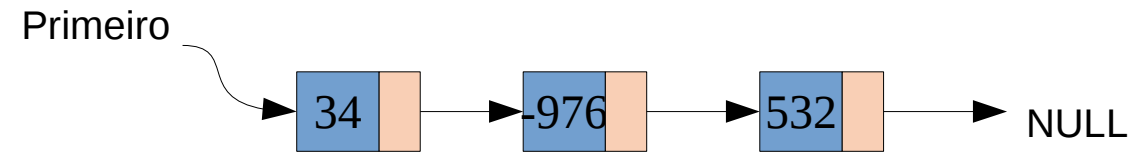
Modifique o programa anterior para imprimir a maior elemento armazenado na lista ligada.

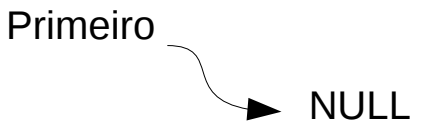
- Nome do arquivo: listaLigada2.c

```
celula* maiorElemento(celula* primeiro); // prototipo


int main() {
    int i, n, numero;

    scanf("%d", &n);
    celula* primeiro = NULL;

    // Leitura dos pontos e criacao dos elementos
    for (i=0; i<n; i++) {
        scanf("%d", &numero);
        primeiro = inserirElemento(primeiro, numero);
    }

    celula* maior = maiorElemento(primeiro);
    printf("\n%d", maior->conteudo);
}
```

listaLigada2.c

```
celula* maiorElemento(celula* primeiro) {
    celula* resposta = primeiro;
    celula* p;

    for (p=primeiro; p!=NULL; p=p->seg) {
        if (resposta->conteudo < p->conteudo)
            resposta = p;
    }
    return resposta;
}
```





listaLigada2.c

```
$ gcc listaLigada2.c -o listaLigada2.exe

$ ./listaLigada2.exe < vetor101.txt
50

$ ./listaLigada2.exe < vetor102.txt
50

$ ./listaLigada2.exe < vetor100k.txt
2147480886
```

Exercício 3

O seguinte programa é uma modificação do programa anterior utilizado para exemplificar como eliminar o elemento que está na primeira posição da lista.

- Nome do arquivo: listaLigada3.c

listaLigada3.c

```c
int main() {
    int i, n, numero;

    scanf("%d", &n);
    celula* primeiro = NULL;

    // Leitura dos pontos e criacao dos elementos
    for (i=0; i<n; i++) {
        scanf("%d", &numero);
        primeiro = inserirElemento(primeiro, numero);
    }

    printf("\n%d", primeiro->conteudo);
    primeiro = eliminarPrimeiroElemento(primeiro);
    printf("\n%d", primeiro->conteudo);

}
```

```c
celula* eliminarPrimeiroElemento(celula* primeiro) {
    celula* p = primeiro;

    primeiro = p->seg;
    free(p);
    return primeiro;
}
```

listaLigada3.c

```
$ gcc listaLigada3.c -o listaLigada3.exe

$ ./listaLigada3.exe < vetor101.txt
-49
-50

$ ./listaLigada3.exe < vetor102.txt
0
1

$ ./listaLigada3.exe < vetor100k.txt
966003015
1390827185
```

Exercício 4

O seguinte programa é uma modificação do programa anterior utilizado para exemplificar como eliminar o elemento que está em uma determinada posição da lista.

- Nome do arquivo: listaLigada4.c

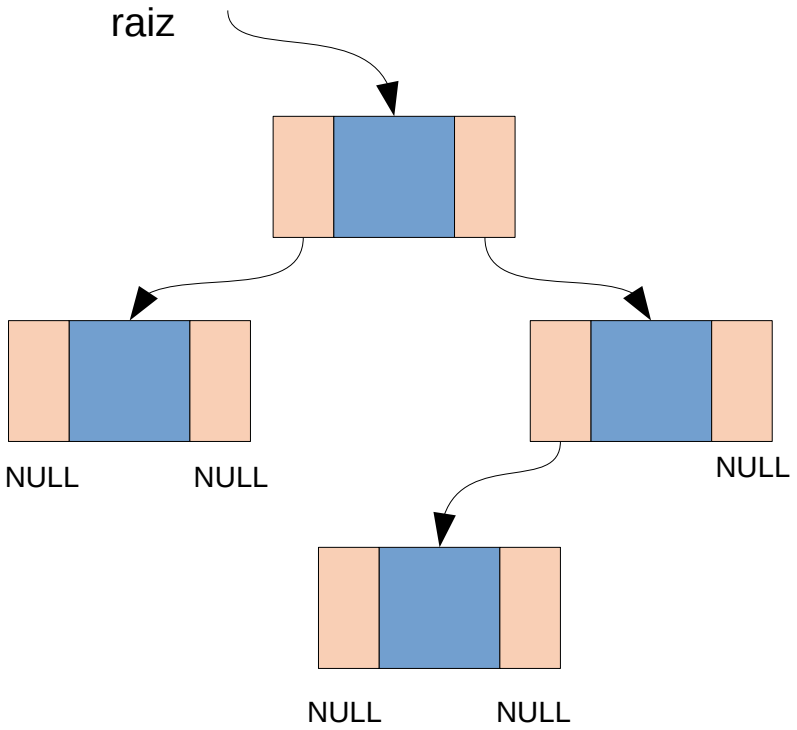
```
celula* eliminarElemento(celula* primeiro, int posicao) {
    celula* p = primeiro;

    if (posicao==0) {
        primeiro = p->seg;
        free(p);
        return primeiro;
    }

    while (posicao>1) {
        posicao--;
        p = p->seg;
    }
    // aqui p->seg aponta ao elemento a ser eliminado
    celula* lixo;
    lixo   = p->seg;
    p->seg = p->seg->seg;
    free(lixo);

    return primeiro;
}
```

Árvore (versão simples)

Uma árvore binária

Uma árvore binária


```c
struct cel {
    int conteudo;
    struct cel *esq;
    struct cel *dir;
};

typedef struct cel celula;
```



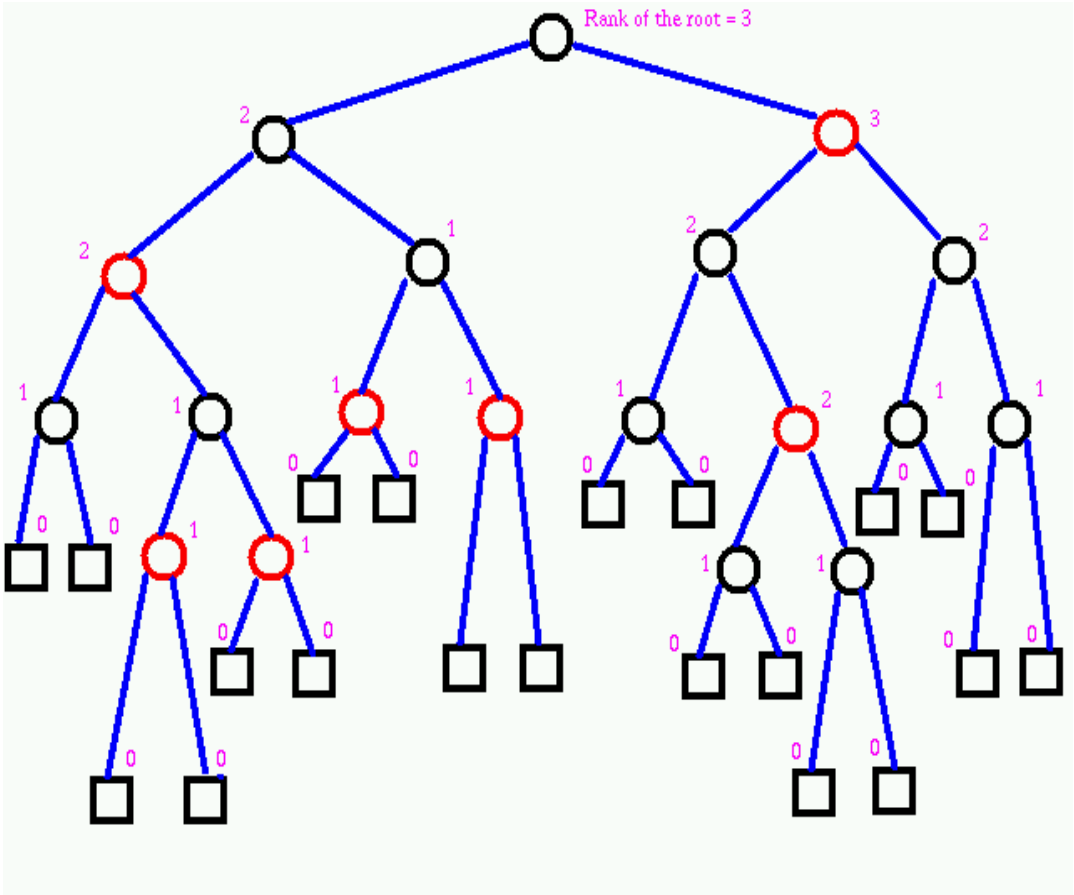
Uma árvore binária

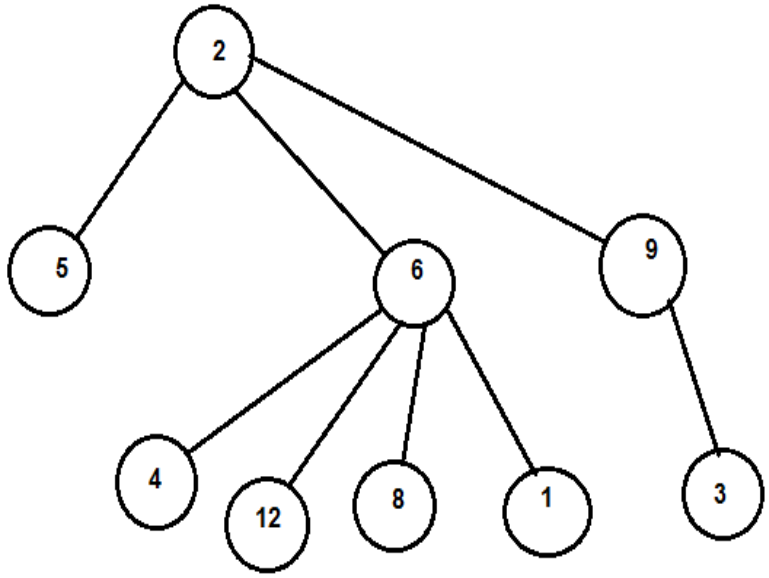

```c
struct cel {
    int conteudo;
    struct cel *esq;
    struct cel *dir;
};

typedef struct cel celula;
```

```c
celula* criarElemento (int numero) {
    celula* novo   = (celula *) malloc(sizeof(celula));
    novo->esq      = NULL;
    novo->dir      = NULL;
    novo->conteudo = numero;

    return novo;
}
```

Nome do arquivo: arvoreBinaria1.c

Árvores





Sobre a Prova 1

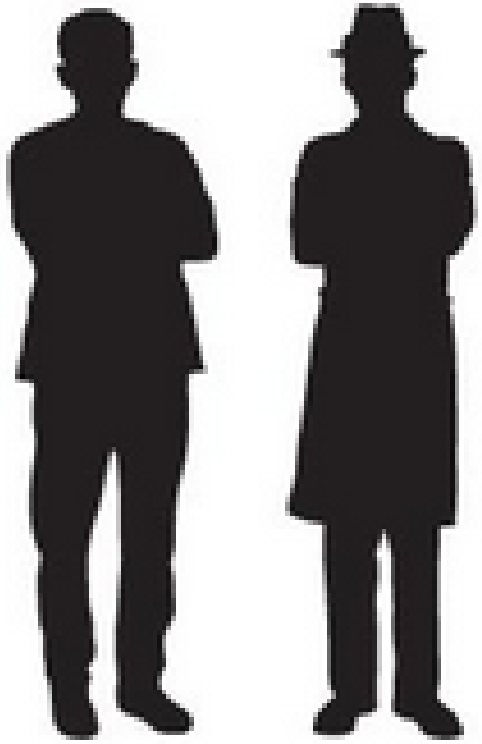
Lorem ipsum dolor sit amet, consectetur adipiscing elit . Curabitur vel est augue. Donec aliquam laoreet ipsum , ac sagittis odio aliquam quis. In eleifend est tortor, dapibus posuere odio consectetur id. Donec varius eget est eu luctus . Phasellus scelerisque, nunc laoreet feugiat sagittis, risus neque condimentum nisi, eget feugiat turpis purus quis urna. Mauris tempor eros in turpis tincidunt, bibendum euismod est pretium. Praesent ligula dui, fermentum a porttitor vitae, pulvinar ut odio. Suspendisse vitae pretium dolor, at sodales mauris .

Sed consequat purus nec bibendum suscipit. Donec ultricies euismod enim, quis interdum mi ornare et . Ut facilisis elit vitae elementum fringilla. Pellentesque sed orci iaculis, imperdiet elit et, mollis enim.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur vel est augue. Donec aliquam laoreet ipsum, ac sagittis odio aliquam quis. In eleifend est tortor, dapibus posuere odio consectetur id. Donec varius eget est eu luctus. Phasellus scelerisque, nunc laoreet feugiat sagittis, risus neque condimentum nisi, eget feugiat turpis purus quis urna. Mauris tempor eros in turpis tincidunt, bibendum euismod est pretium. Praesent ligula dui, fermentum a porttitor vitae, pulvinar ut odio. Suspendisse vitae pretium dolor, at sodales mauris.

Sed consequat purus nec bibendum suscipit. Donec ultricies euismod enim, quis interdum mi ornare et. Ut facilisis elit vitae elementum fringilla. Pellentesque sed orci iaculis, imperdiet elit et, mollis enim.

elegante?



Direitos de autor: Dezignus

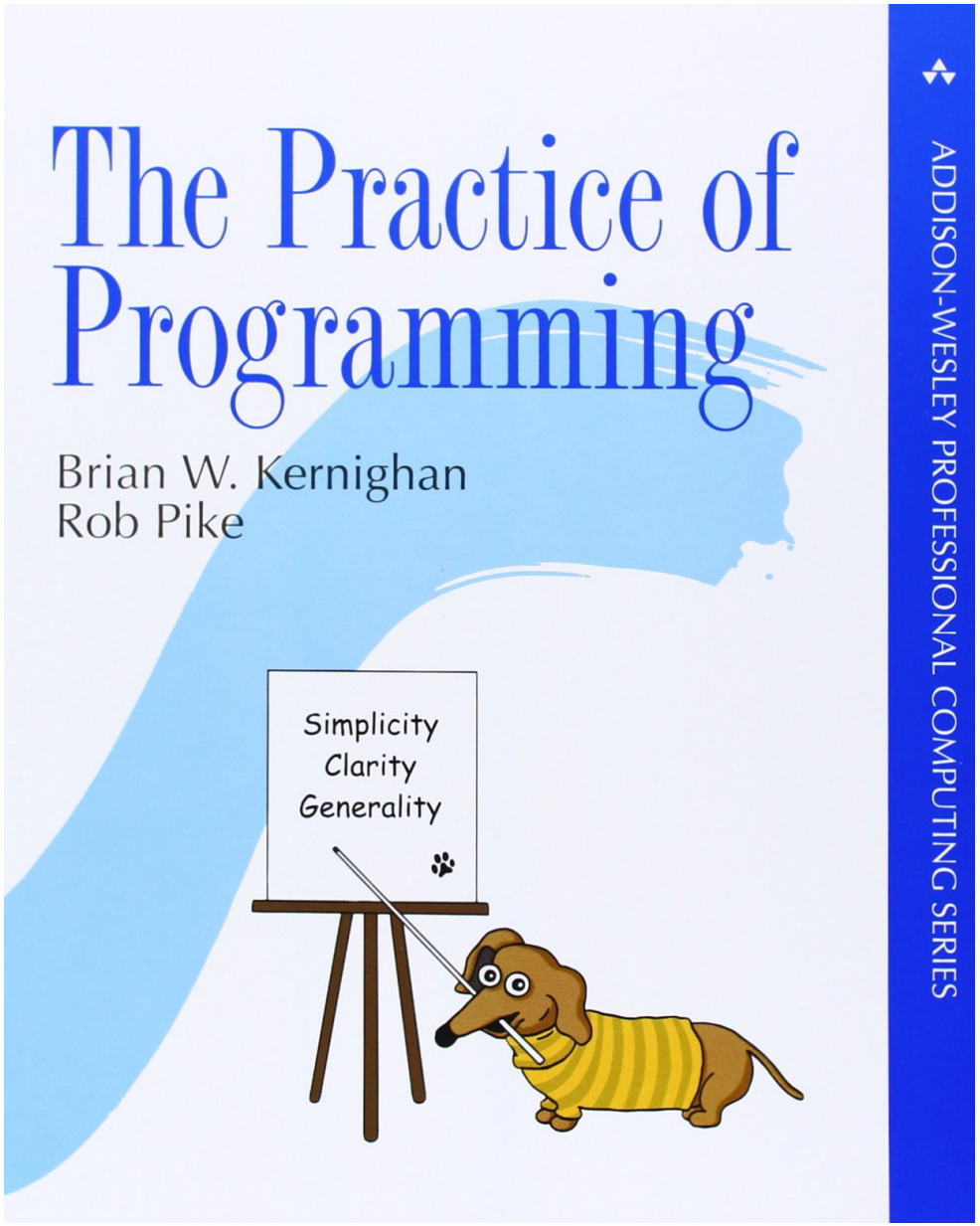
elegante?

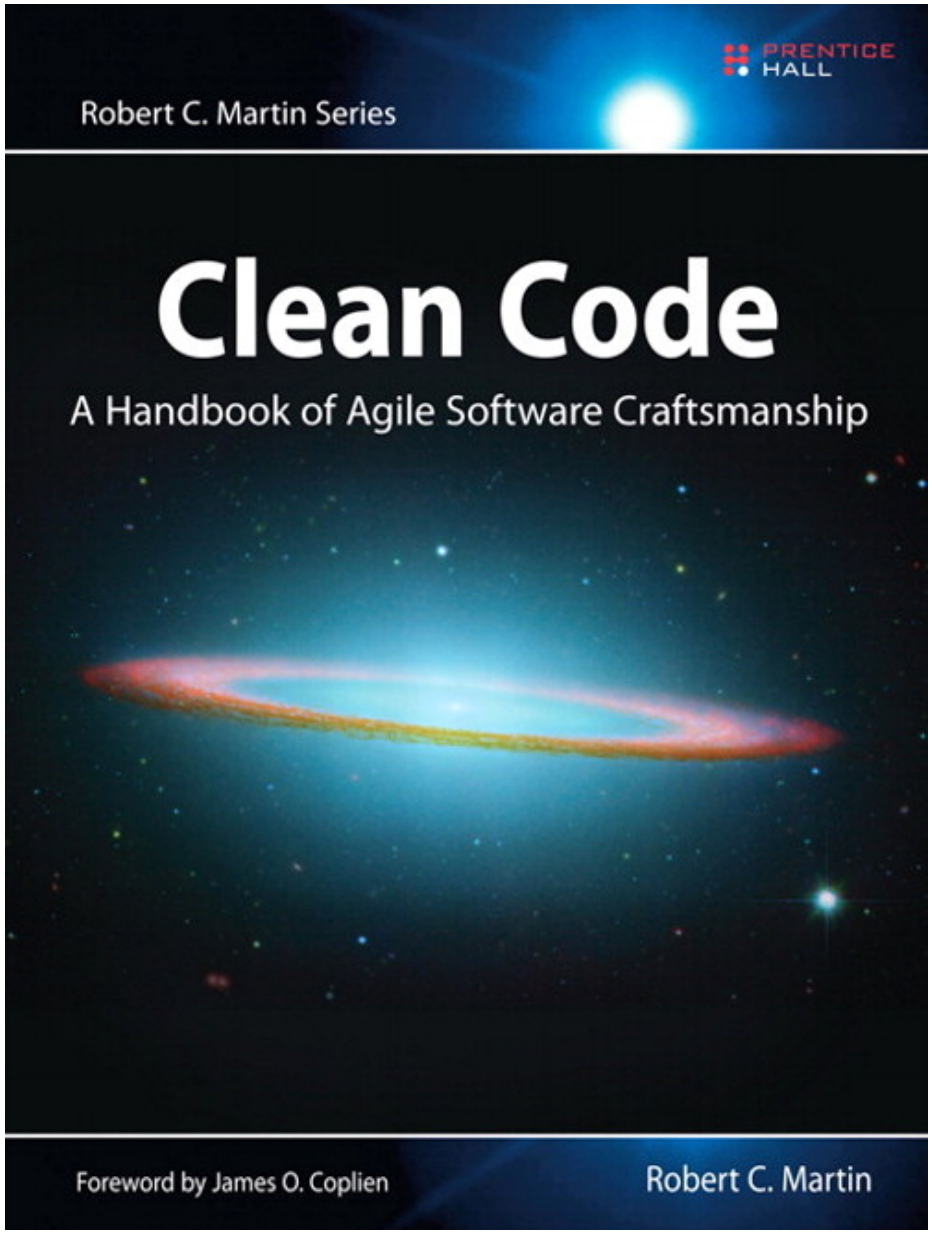
```c
#include <stdio.h>

int main(){
  int n, temp;
  scanf("%d", &n);
  int a[n];


  for(int i = 0; i<n;i++){
    scanf("%d", &a[i]);
    temp = 1;
    for(int j = 1; j<=a[i]; j++){
      if(j%2 == 0) temp++;
      else temp*=2;
    }
    a[i] = temp;
  }

  for(int i=0; i<n; i++){
    printf("%d\n", a[i]);
  }


return 0;
}
```

```c
#include <stdio.h>

int main() {
    int i, j, n, temp;
    scanf("%d", &n);
    int a[n];

    for(i=0; i<n; i++) {
        scanf("%d", &a[i]);
        temp = 1;
        for(j=1; j<=a[i]; j++) {
            if(j%2 == 0)
                temp++;
            else
                temp*=2;
        }
        a[i] = temp;
    }

    for(i=0; i<n; i++) {
        printf("%d\n", a[i]);
    }

    return 0;
}
```

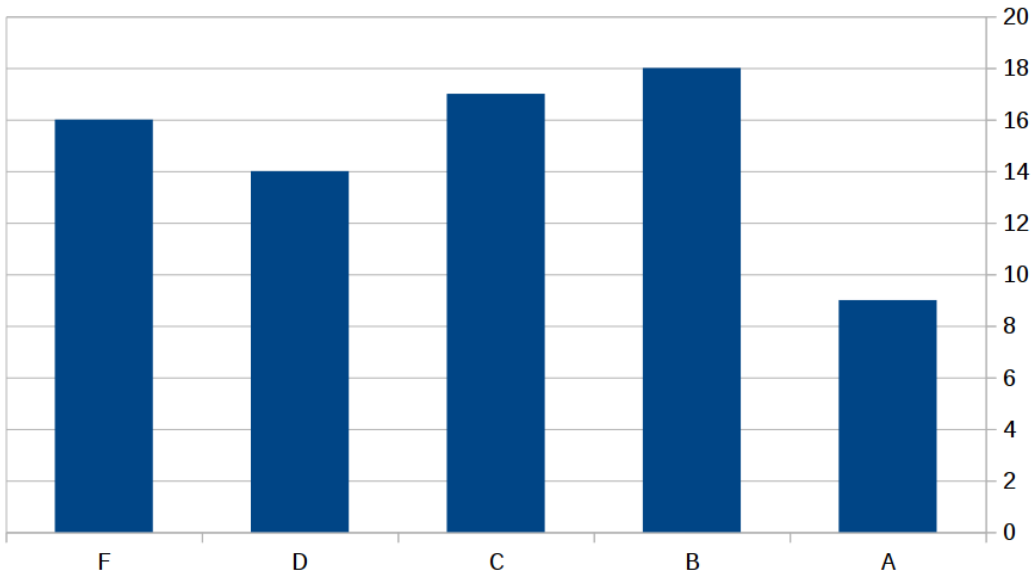
elegante?

```
1  #include <iostream>
2  #include <stdlib.h>
3
4
5
6  using namespace std;
7
8  int main() {
9
10 string sinal, paridade;
11 int n;
12
13 scanf("%d", &n);
14 int *v = (int *) malloc(n * sizeof(int));
15
16 for (int i=0; i<n; i++){
17 scanf("%d", v+i);
18 }
19
20 for (int i=0; i<n; i++){
21 if(*(v+i)%2==0)
22 paridade = "EVEN";
23 else
24 paridade = "ODD";
25 if(*v<0)
26 sinal = "POSITIVE";
27 else
28 sinal = "NEGATIVE";
29
30 if(*(v+i)==0)
31 printf("NULL");
32 else
33 printf("%s %s\n",paridade, sinal);
34
35 }
36
37     return 0;
38 }
```


The Practice of Programming
Brian W. Kernighan
Rob Pike
Simplicity
Clarity
Generality
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


Robert C. Martin Series
PRENTICE HALL
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin

Prova 1



A	9	12.16%
B	18	24.32%
C	17	22.97%
D	14	18.92%
F	16	21.62%