Sub-rotinas

Aula 16

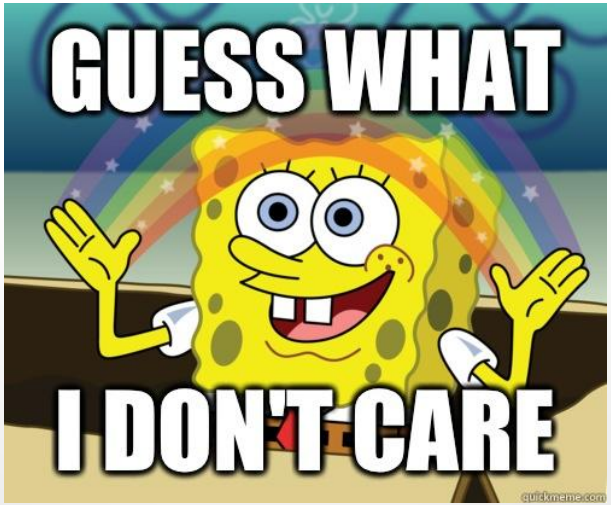
Às vezes, só falta organizar...

O que é uma sub-rotina

- Uma sub-rotina é uma sequência de instruções organizada para facilitar o reuso
- Ajuda a pensar um problema maior em pedaços menores
- Mini-programas que resolvem um problema específico

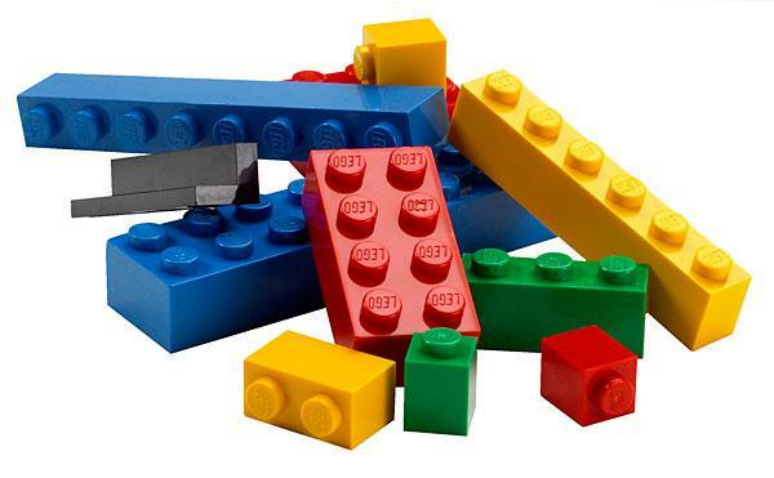
Velhas conhecidas….

- Como funciona a função scanf? E a função printf?
  - Não importa!! Apenas precisamos saber como usar!



Sub-rotinas

- Encapsulamento do funcionamento
- Modularização do programa
- Podem ser acionadas em qualquer ponto do programa



Estrutura de uma sub-rotina

- Uma sub-rotina possui
  - Nome
  - Conjunto de parâmetros (opcional)
  - Conjunto de instruções associadas em um bloco
  - Valor de retorno de suas operações (opcional)

Exemplo

```
int maior(int x, int y){
   if(x>y)
      return x;
   else
      return y;
}
```

Tipos de Sub-rotina

- Um procedimento executa uma sequência de instruções e termina

- Uma função executa uma sequência de instruções e retorna um valor para seu ponto de chamada
  - Em C tudo é função! Para fazer um procedimento, usamos um retorno de tipo void

Exemplo - Procedimento

```
void imprime_letra(char letra, int qtd){
   int i;
   for(i=0; i<qtd; i++){
      printf("%c", letra);
   }
   return; //Ou poderia nem colocar!
}
```

Exemplo - Função

```
int potencia(int x, int y){
   int i;
   int pot = 1;
   for(i=0; i<y; i++){
      pot *=x;
   }
   return pot;
}
```

Declarando a função

- Em C, cada função tem nome único
  - Não existe sobrecarga de função

- A função pode ser declarada em qualquer parte do corpo do programa

- O comando return é usado para retornar um valor da função

Assinatura da função

- Chamamos de assinatura da função o conjunto formado pelo nome e pelos parâmetros da função

- Também chamado de interface da função, explicita como devemos usá-la

- Ao criarmos uma biblioteca de funcionalidades para outros usarem, devemos ter
  - um arquivo .h com as assinaturas das funções (interfaces)
  - um arquivo .c com as implementações das funções

Exemplos

```
int primos(int qtd);

void atualiza_dados(double x, int y);

char inverte_letra(int codigo);
```

Como acontece por baixo

- Quando é chamada, a função ocupa uma área da memória específica contendo todas as suas variáveis

- Uma função apenas enxerga as variáveis que fazem parte dessa área da memória. Isso é chamado de escopo da função

Escopo da função

- As operações da função afetam apenas valores que fazem parte do seu escopo, não interferindo com os valores do programa que executou a função
  - Passagem por valor

- Em C, podemos definir variáveis chamadas globais, que são visíveis a todas as funções da aplicação

Exemplo

```
int soma(int x){
    x +=1;
    printf(“%d”, x) //imprime 2
    return x;
}
```

```
int main(){
    int x, y;
    x = 1;
    y = soma(x);
    printf(“%d”, x); //imprime 1
    printf(“%d”, y); //imprime 2
}
```

Treinando

```
void misterio( int *p, int *q ){
   int r = *p;
   *p = *q;
   *q = r;
}

int main ( ){
   int i = 1;
   int j = 2;
   misterio( &i, &j );
   printf( "%i %i\n", i, j );
   return 0;
}
```

Dúvidas

Problema 01

- Itamir finalmente conseguiu realizar seu sonho de explorar as selvas africanas! Por ser um expert em Android, ele decidiu implementar um app Bússola para guiá-lo pela floresta!

Welcome to the Jungle

- Só que Itamir esbarrou em um problema: a sua bússola não está mapeando as direções corretamente quando ele se movimenta! E ele quer a ajuda de vocês para consertar.

Welcome to the Jungle

- Escreva uma função que, dada quatro variáveis e uma movimentação, atualize a posição dele. As variáveis representarão a posição de Itamir, e conterão as quatro direções atuais. Elas deverão ser atualizadas de acordo com a movimentação feita.

Welcome to the Jungle

Direções:

N = Norte
S = Sul
L = Leste
O = Oeste

Movimentações:

A = Andar para frente
D = Virar 90º para direita
E = Virar 90 º para esquerda
V = Virar 180º

Welcome to the Jungle

Entrada:
Frente = N
Atrás = S
Direita = L
Esquerda = O
Movimentação = D

Saída
Frente = L
Atrás = O
Direita = S
Esquerda = N

Programa Base

```
int main(){
     char frente, atras, direita, esquerda;
     char movimento;
     scanf(“%c %c %c %c”, &frente, &atras, &direita, &esquerda);
     while(1){
          scanf(“%c %c”, &movimento, &movimento);
          if(movimento == ‘F’)
               break;
          else
               ajusta_bussola(&frente, &atras, &direita, &esquerda, movimento);
     }
     printf(“Posição final:\nFrente: %c\nDireita: %c\nAtrás: %c\nEsquerda: %c\n”,
frente, direita, atras, esquerda);
     return 0;
}
```

Proposta de solução

```
void ajusta_bussola(char *f, char* a, char* d, char* e, char
mov){
    char temp;

    switch(mov){
        case 'A':
            break;
```

Proposta de Solução

```
case 'D':
            temp = *f;
            *f = *d;
            *d = *a;
            *a = *e;
            *e = temp;
            break;
```

Proposta de Solução

```
case 'E':
            temp = *f;
            *f = *e;
            *e = *a;
            *a = *d;
            *d = temp;
            break;
```

Proposta de Solução

```
case 'V':
            temp = *f;
            *f = *a;
            *a = temp;

            temp = *d;
            *d = *e;
            *e = temp;
            break;
    }
}
```

Exercícios

- Escreva uma função que receba um valor inteiro e retorne o seu valor ao quadrado

Entrada:	Saída:
2	4
4	16
9	81

Exercícios

- Escreva uma função que recebe três números inteiros e retorna o maior deles

Entrada	Saída
3 4 5	5
45 70 2	70
54 34 12	54

Exercícios

Escreva um programa que leia três números e retorne o quadrado do maior deles

Entrada	Saída
3 4 5	25
10 8 2	100
6 12 4	144