UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Fundamentos de Programação

CP41F

Estruturas de iteração (for, while, do-while). Controle de laços (break, continue). Depuração.

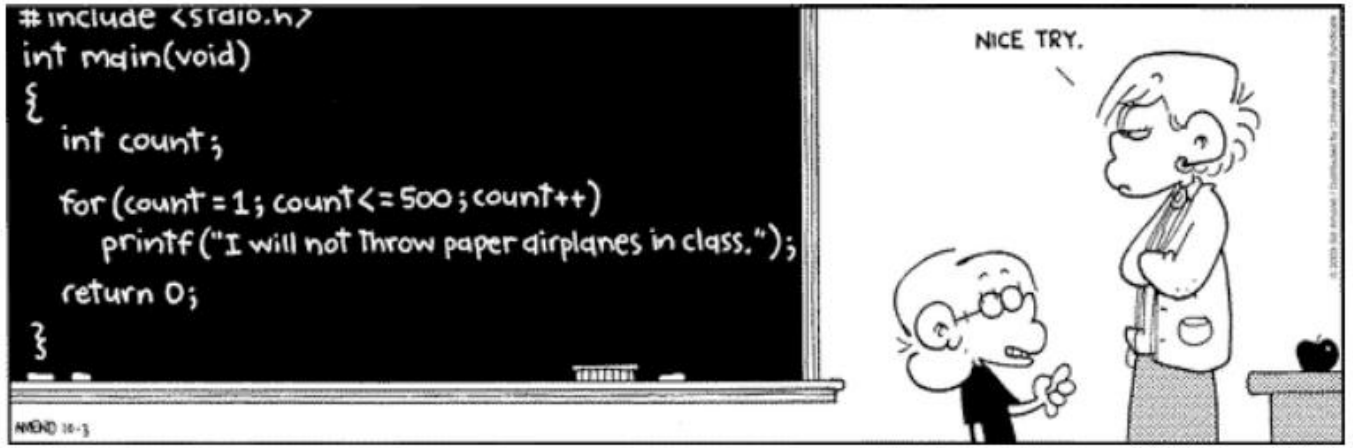
Aula 10

Prof. Daniel Cavalcanti Jeronymo

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação – 1º Período

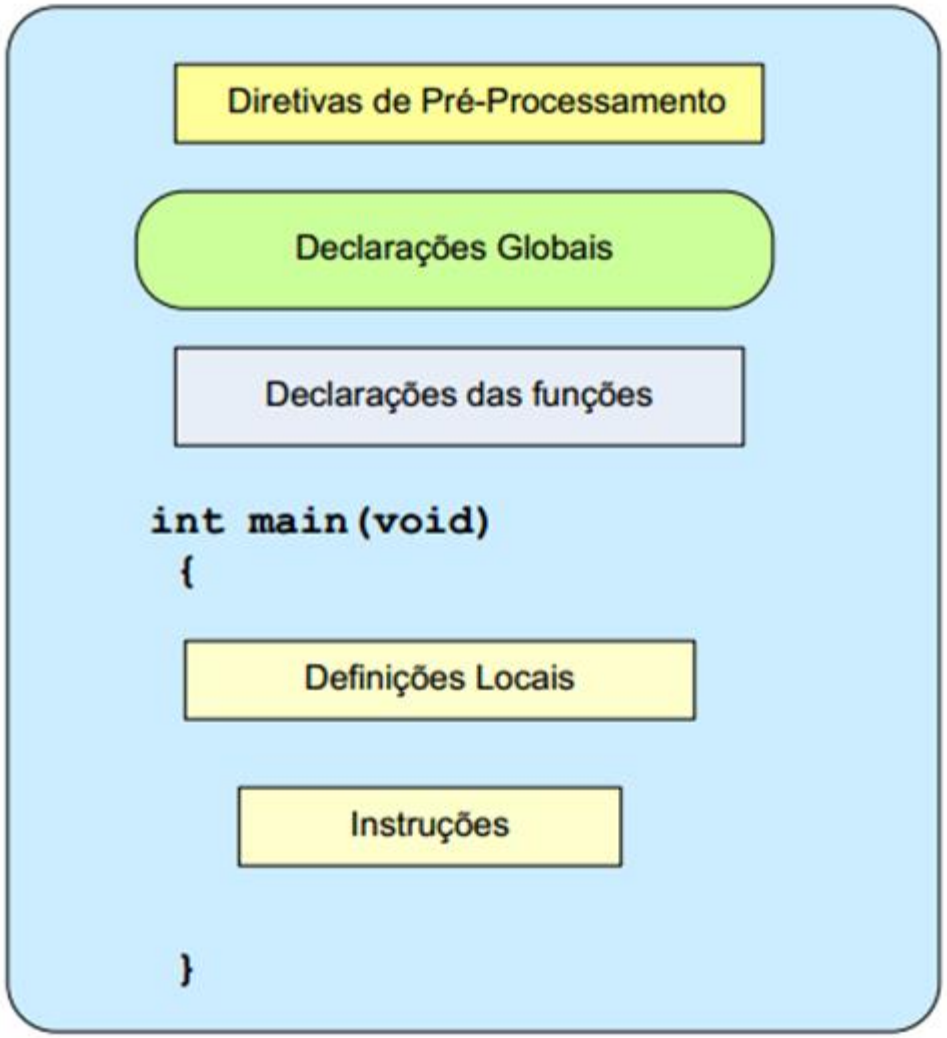
Plano de Aula

- Revisão de programação estruturada
  - declaração, seleção e iteração
- Estruturas de iteração
  - while, do-while, for
- Controle de laços
  - break, continue
- Depuração (debug)



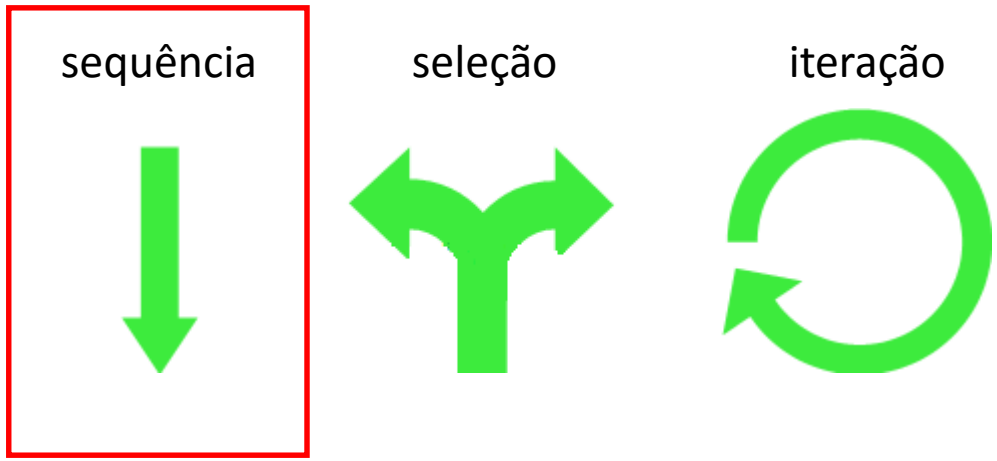
Revisão

- Blocos
  - Agrupamento de declarações
  - Definição de escopo
  - Delimitado por chaves { }

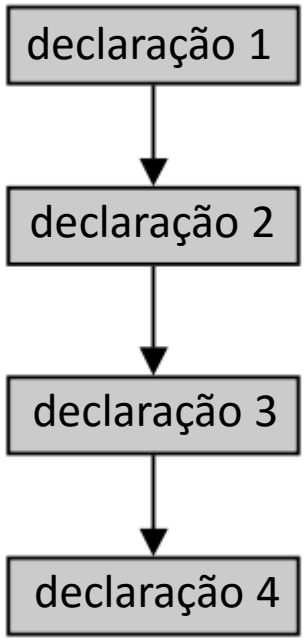


Revisão

- Programação Estruturada
  - Sequência: ordenação de declarações **sequenciais**

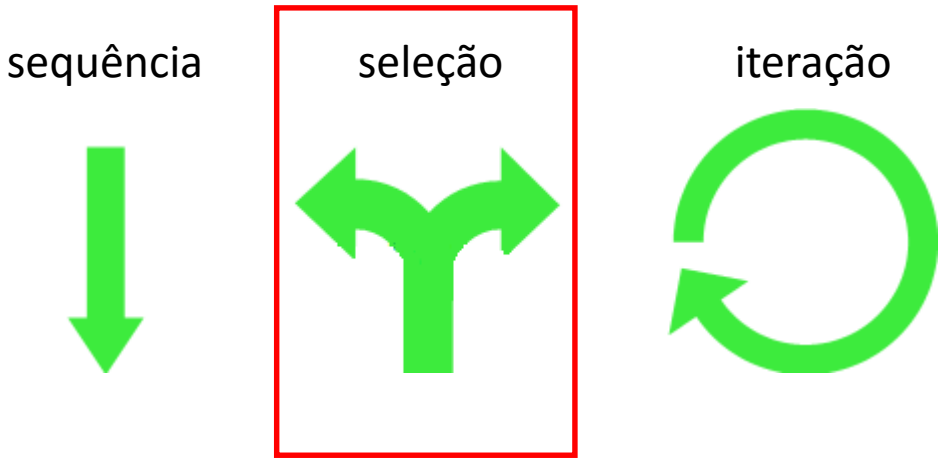



```
declaração 1;
declaração 2;
declaração 3;
declaração 4;
```

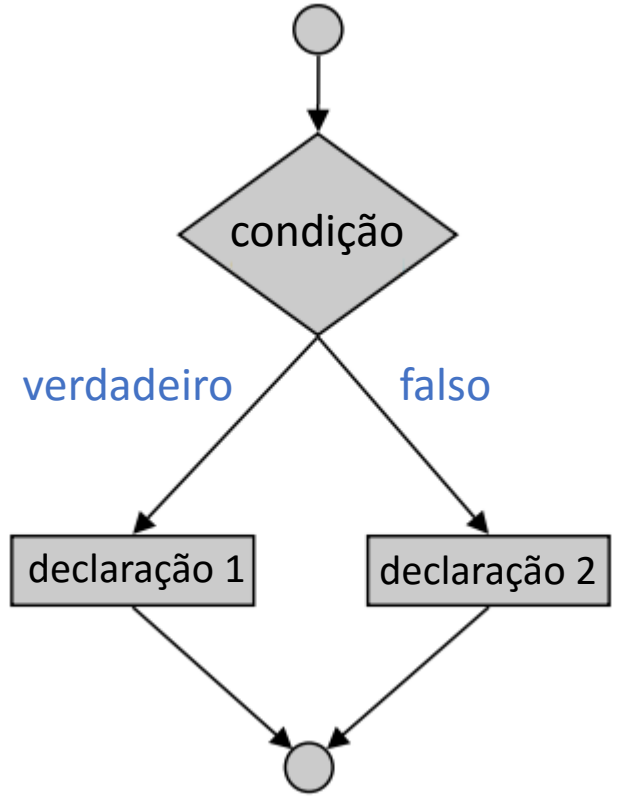


Revisão

- Programação Estruturada
  - Seleção: controle de fluxo para **ramificação**
    - Verifica uma condição lógica
    - **SE** for verdadeira, executa um bloco
    - **SENÃO**, executa outros blocos




```
if(condição)
   declaração 1;
else
   declaração 2;
```



Revisão

- Programação Estruturada
  - Iteração: controle de fluxo para **repetição**

- Verifica uma condição lógica
- **SE** for verdadeira, executa um bloco
- **SENÃO**, continua

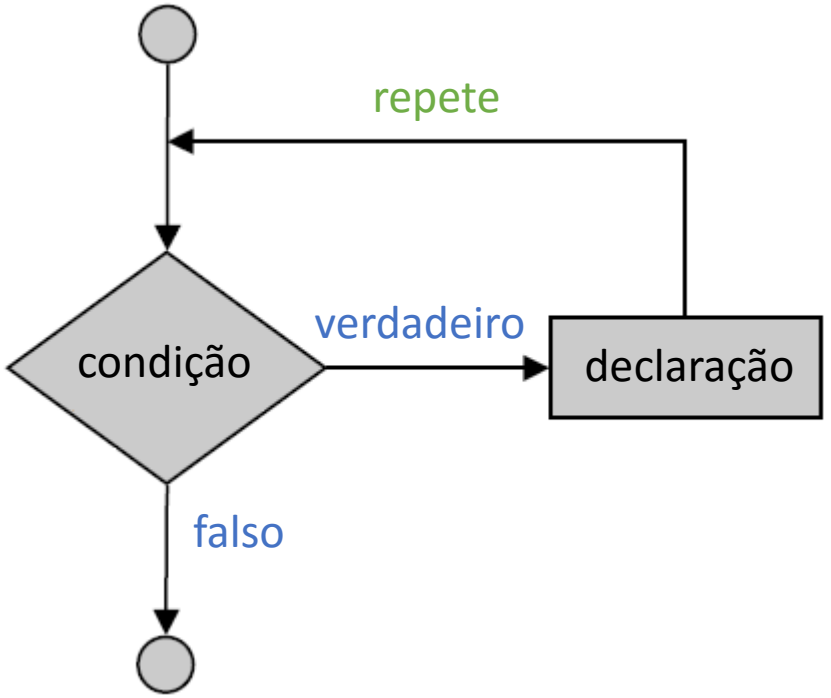
repete

sequência

seleção

iteração

```
while(condição)
declaração;
```

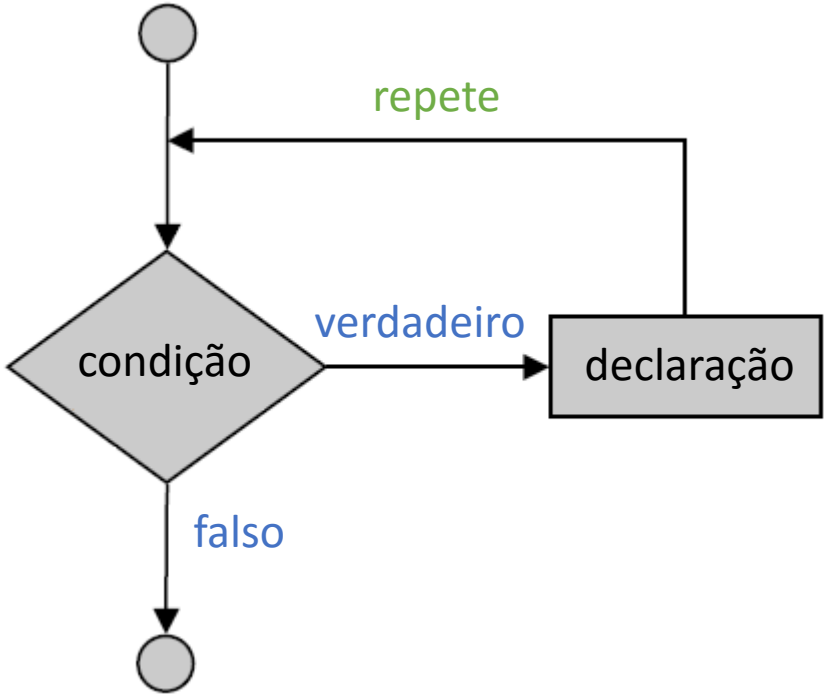


Estruturas de iteração

- Estruturas de iteração
  - *while*
  - *do - while*
  - *for*

Estruturas de iteração

- while
  - **ENQUANTO** a condição é verdadeira, faça o seguinte...



Estruturas de iteração

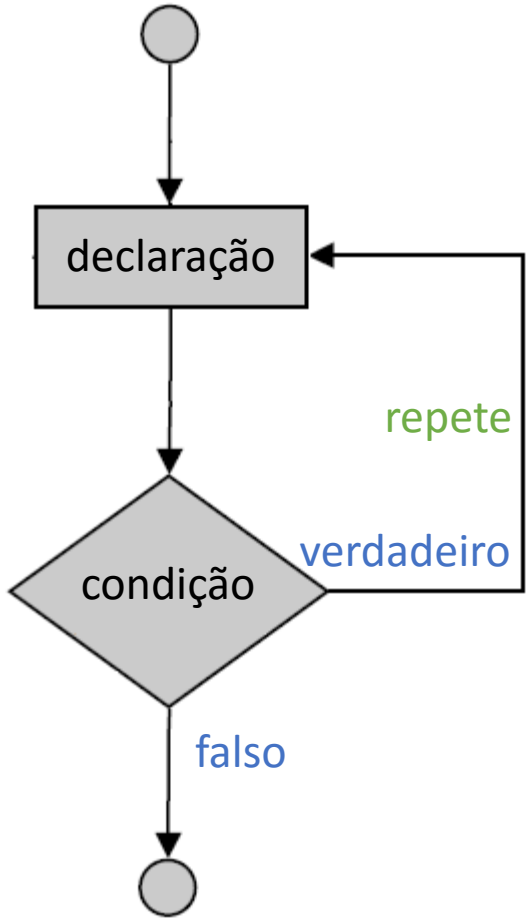
- while
  - Exemplo: potências sucessivas de dois

```c
#include <stdio.h>

int main()
{
   int vezes = 2;
   int valor = 2;
   int i = 0;

   while(i < vezes)
   {
      valor *= 2;
      printf("%d\n", valor);

      i++;
   }

   return 0;
}
```

Como verificar a lógica?
Teste de tabela!

Estruturas de iteração

- do - while
  - **FAÇA** o seguinte ... **ENQUANTO** a condição é verdadeira



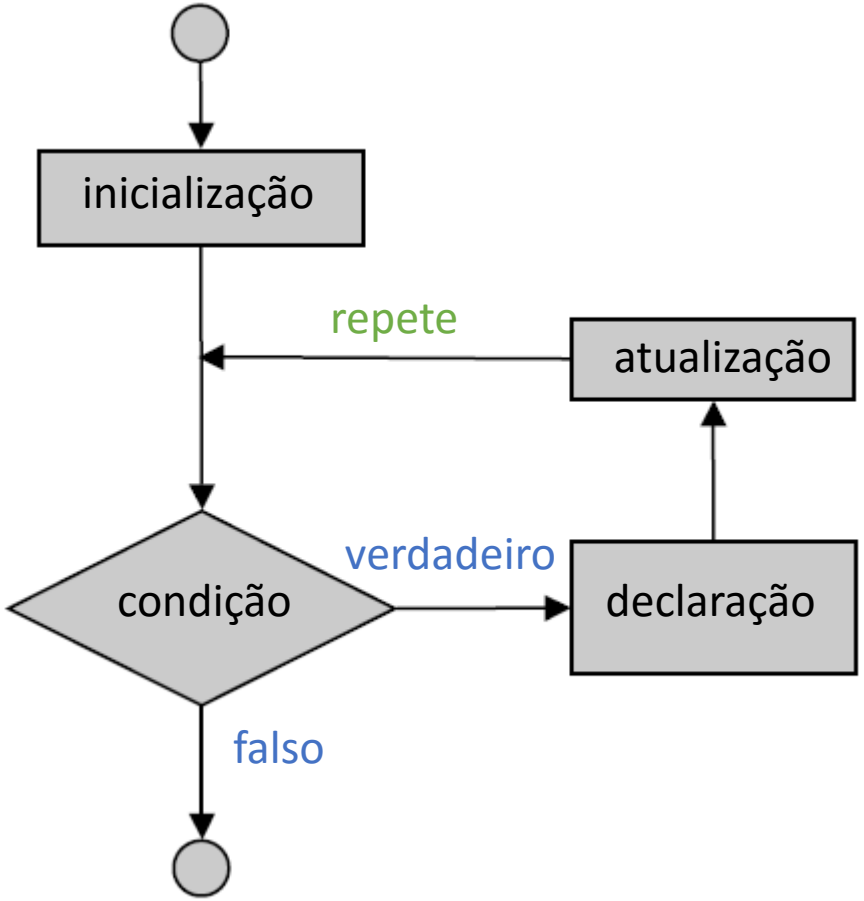
Estruturas de iteração

- do - while
  - Exemplo: entrada de dados

```
#include <stdio.h>

int main()
{
   char c = 0;

   do
   {
      printf("Observe o lixo na entrada - entre
com 'S' para sair\n");
      scanf("%c", &c);
   }
   while (c != 'S');

   return 0;
}
```

Estruturas de iteração

- for
  - **FAÇA** N vezes



Estruturas de iteração

- for
  - Exemplo: imprimir valores de 0 a 9

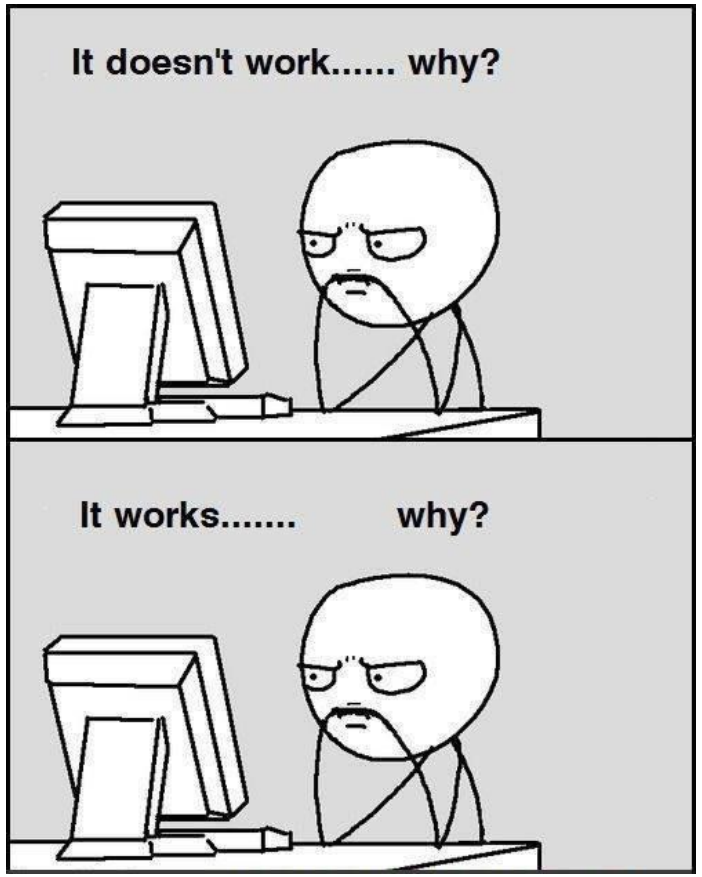
```
#include <stdio.h>

int main()
{
   int i;

   for(i=0; i < 10; i++)
      printf("%d\n", i);

   return 0;
}
```

Estruturas de iteração

- Quando usar cada um?
  - *for* – quando sabe-se quantas vezes um laço será executado, i.e. para iterar entre elementos
  - *while* – quando não sabe-se quantas vezes um laço será executado
  - *do-while* – quando não sabe-se quantas vezes um laço será executado, porém há a necessidade de executá-lo no mínimo uma vez

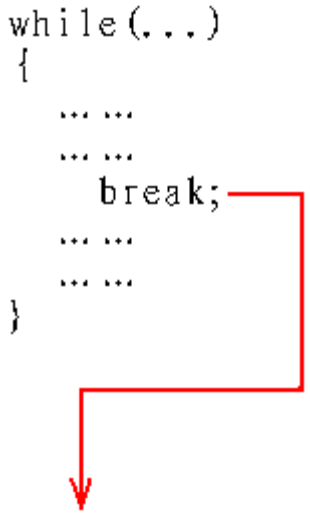
Estruturas de iteração

- Equivalências
  - É possível reescrever todas as estruturas de iteração em função de *while*?



Controle de laços

- **break**
  - Sai imediatamente do bloco interior em estruturas *for*, *while*, *do-while* ou *switch*
  - A execução continua a partir do fim do bloco



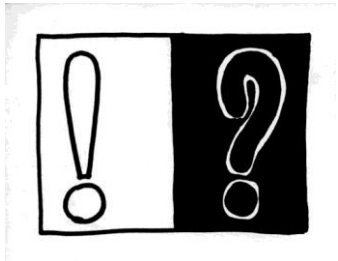
  - Usos comuns:
    - Sair de um laço
    - Ignorar o restante de um switch

Controle de laços

- **break**
  - Qual a saída do seguinte código?

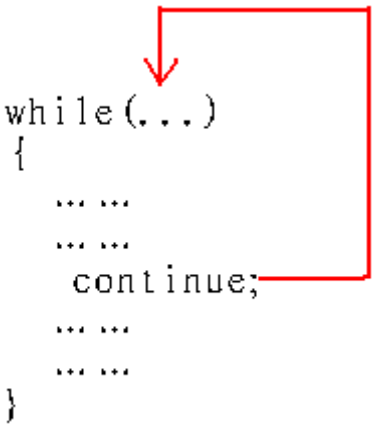
```
#include <stdio.h>

int main()
{
   while(1)
   {
      printf("Linha 7\n");
      while(1)
      {
         printf("Linha 10\n");
         break;
      }
   }

   return 0;
}
```

Como sair do laço externo a partir do laço interno?



Controle de laços

- **continue**
  - Pula o restante das declarações em um bloco de *while*, *for* ou *do-while*
  - Continua com a próxima iteração do laço
  - *while* e *do-while*
    - A condição é avaliada imediatamente após a declaração continue
  - *for*
    - A expressão de incremento é executada, em seguida a condição é avaliada



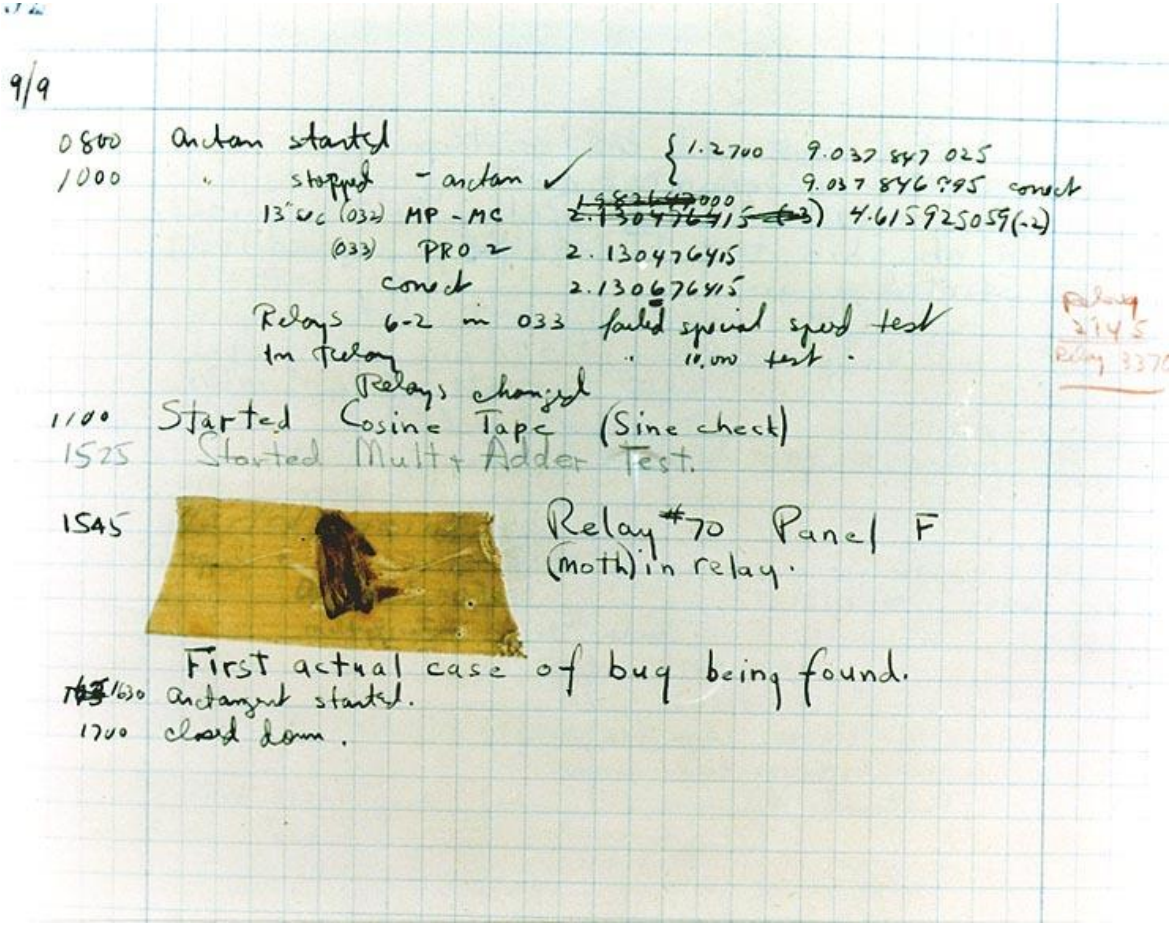
Controle de laços

- **continue**
  - Qual a saída do seguinte código?

```
#include <stdio.h>

int main()
{
   while(1)
   {
      printf("Linha 7\n");
      while(1)
      {
         printf("Linha 10\n");
         continue;
      }
      break;
   }

   return 0;
}
```

Depuração

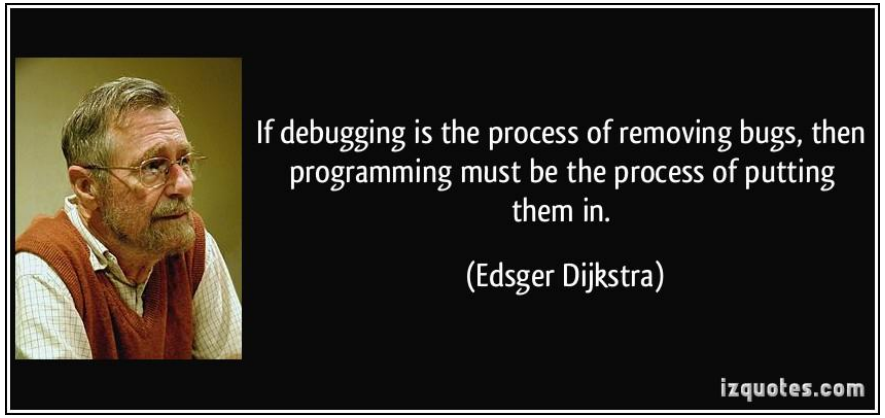
- Bugs
  - Primeiro caso: mariposa presa em um relé num Harvard Mark II



Grace Hopper, US Navy

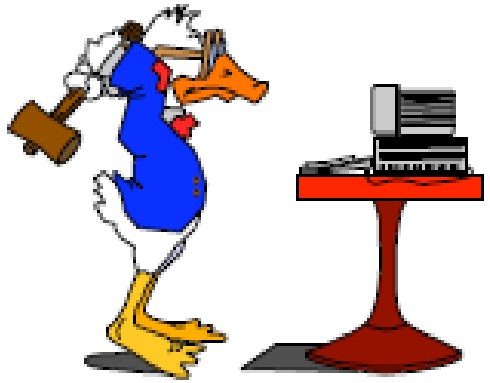
Depuração

- Bugs
  - Erros de compilação: mensagens de erro ajudam a localizar o problema
  - Erros de semântica/lógica: não há indicador claro, precisam ser depurados



Depuração

- Depuração (debugging)
  - Processo cíclico de localização, edição, compilação e verificação
  - Sempre há uma explicação
  - O que a máquina faria?



Depuração

- Depuração (debugging) – alternativas principais
  - Tracing – marcação em pontos do código para rastrear o fluxo de controle (*printf debugging*)
  - Quando o problema é mais complexo, use um debugger!