Entrada e Saída de Dados

Linguagem de Programação I

Waldson Patrício

Introdução

- Entrada e Saída Padrão do C++
- `argc` e `argv`
- `return 0`
- `getline()`

Dispositivos de Entrada e Saída

Entrada

- Teclado
- Mouse
- Joystick
- …

Saída

- Monitor
- Fone de Ouvido
- Impressoras
- …

Dispositivos Híbridos e Não Convensionais

- ???

Dispositivos de Entrada e Saída

Entrada

- Teclado
- Mouse
- Joystick
- …

Saída

- Monitor
- Fone de Ouvido
- Impressoras
- …

Dispositivos Híbridos e Não Convensionais

- Joystick, Telas *Touch*, Sensores, Internet

Entrada e Saída Padrão do C++

- O cabeçalho `<iostream>` contém **objetos** `stream` responsáveis pela entrada (`input`) e saída (`output`) padrões:
  - `std::cin`
  - `std::cout`
  - `std::cerr`
  - `std::clog`
- A entrada e saída padrões em um programa C++ vem do terminal

Entrada e Saída Padrão do C++

- Os `streams` são uma abstração em cima de dispositivos e representações de *coisas* que podem servir como entrada e saída:
  - `STDIN` e `STDOUT`
  - Arquivos
  - Um objeto `std::string`
  - Rede
  - Os dispositivos citados anteriormente

Entrada e Saída Padrão do C++

- Os operadores de *bitshift* ganham uma nova função quando estamos tratando com *streams*:
  - >> operador de extração (*extraction operator*)
    - Recurepa um dado de um *stream* e coloca em uma variável
  - << operador de inserção (*insertion operator*)
    - Adiciona um valor a um *stream*

`argc` e `argv`

- São os argumentos de linha de comando do seu programa
- **argc** indica a quantidade de argumentos (*argument count*) que o seu programa recebeu
  - O primeiro argumento é sempre o nome do programa
- **argv** é um array contendo todos os argumentos
  - Todos os argumentos são *strings* C.
  - Os argumentos são delimitados por espaços em branco ou quebra de linha se não estiverem cercados por aspas

Quando usar `std::cin` e quando usar argumentos de linha de comando?

- Geralmente usar argumentos de linha de comando dão mais flexibilidade ao programa
- Usamos *std::cin* quando queremos que o programa seja interativo

`return 0`

- O retorno da função `main` define o status de saída do seu programa:
  - Qualquer valor diferente de 0 indica um erro na execução
  - Em C há duas constantes que representam isso:
    - `EXIT_SUCCESS` (0)
    - `EXIT_FAILURE` (1)

Pode ser considerado como uma saída do seu programa:

- Informa se o seu programa executou ou não o que deveria

`getline()`

- `std::getline(std::cin, std::string)` extraí uma linha completa do `stream` de entrada

Prática

- Criar um programa que recebe como argumentos de linha de comando a seguinte ação:
  - `add` Esta ação verifica se há outro parâmetro de linha de comando:
    - Se houver ele imprime `Mensagem adicionada`
    - Se não houver, ele solicita ao usuário um parâmetro, e depois imprime `Mensagem Adicionada`
- Se não receber nenhum parâmetro deve imprimir seu uso

Prática

- Exemplos de Entrada e suas respectivas saídas
  - `./prog`
    - Saída: `Uso:  ./prog add <mensagem>`
  - `./prog add`
    - Solicita ao usuário uma mensagem
    - Imprime `Mensagem Adicionada`
  - `./prog add mensagem`
    - Imprime `Mensagem Adicionada`

Presença de Hoje

- Link para este código enviado via Discord no canal `#presença` em um repositório seu no Github ou no servidor do IMD
  - https://projetos.imd.ufrn.br
  - https://github.com