fit@hcmus

Files

1

fit@hcmus

Console Input/Output

3

fit@hcmus

Console Input/Output

- Using these objects: `std::cin`, `std::cout`, `std::cerr` of `iostream`
- Declaring before use:

```
#include <iostream>
```


4

fit@hcmus

Input Using `std::cin`

- `std::cin` is used with >> to gather input
- The stream **extraction operator** is >>
- Using more than one variable in `std::cin` allows more than one value to be read at a time
- Examples:
  - `std::cin >> miles;`
  - `std::cin >> numberofLanguages;`
  - `std::cin >> dragrons >> trolls;`
  - `std::cin >> dragrons >> trolls;`

5

fit@hcmus

Input Using `std::cin`

- `std::in` stops when getting white spaces.
- Try to use function `getline` of `std::cin`.

6

fit@hcmus

Input Using `std::cin`

```
#include <iostream>
#include <cstring>
int main(){
      char name[80];
      std::cout << "Input your name: ";
      std::cin.getline(name, 80);
      std::cout << "Your name is " << name << "\n";
      return 0;
}
```


7

Output Using `std::cout`

- Any combinations of variables and strings can be output.
- `std::cout` is used with << to output.
- The stream **insertion operator** is <<
- Expression evaluated and its value is printed at the current cursor position on the screen.

8

Output Using `std::cout`

- The new line character is '`\n`' . May appear anywhere in the string.
- `std::endl` causes insertion point to move to beginning of next line.

9

fit@hcmus

Output Using `std::cout`

- Commonly used escape sequences:

	Escape Sequence	Description
`\n`	Newline	Cursor moves to the beginning of the next line
`\t`	Tab	Cursor moves to the next tab stop
`\b`	Backspace	Cursor moves one space to the left
`\r`	Return	Cursor moves to the beginning of the current line (not the next line)
`\\`	Backslash	Backslash is printed
`\'`	Single quotation	Single quotation mark is printed
`\"`	Double quotation	Double quotation mark is printed

10

fit@hcmus

Output Using `std::cout`

- Formatting for Numbers:

```
std::cout.setf(std::ios::fixed);
std::cout.setf(std::ios::showpoint);
std::cout.precision(2);
```

```
std::cout.setf(std::ios::fixed);
std::cout.setf(std::ios::showpoint);
std::cout.precision(2);

std::cout << 7.9999 << " " << 10.5 << std::endl;
```

```
8.00 10.50
```


11

fit@hcmus

Output Using `std::cout`

```
#include <iostream>
#include <iomanip>
int main()
{
    double pi_v = 3.14159, npi = -3.14159;

    std::cout << std::fixed << std::setprecision(0) << pi_v << " " << npi << std::endl;
    std::cout << std::fixed << std::setprecision(1) << pi_v << " " << npi<< std::endl;
    std::cout << std::fixed << std::setprecision(3) << pi_v << " " << npi << std::endl;
    std::cout << std::fixed << std::setprecision(4) << pi_v << " " << npi << std::endl;
    std::cout << std::fixed << std::setprecision(5) << pi_v << " " << npi << std::endl;
    std::cout << std::scientific << std::setprecision(6) << pi_v << " " << npi << std::endl;

    return 0;
}
```

```
3 -3
3.1 -3.1
3.142 -3.142
3.1416 -3.1416
3.14159 -3.14159
3.141590e+00 -3.141590e+00
```


12

fit@hcmus

File

- A file is a container in a computer system for **storing information**.
- There are different types of files such as text files, data files, directory files, binary and graphic files, etc.
- Files can be stored on optical drives, hard drives or other types of storage devices.

(*techopedia)*

13

fit@hcmus

File

- Categorization:
  - Text files
  - Binary files
- File name:
  - Name
  - File extension
- Filepath
  - Absolute path
  - Relative path

14

fit@hcmus

File

- The **basic operations** that can be performed on a file are:
  - Creation of a new file
  - Modification of data or file attributes
  - Reading of data from the file
  - Writing data to the file
  - Closing or terminating a file operation

15

Reading from a Text File

fit@hcmus

- using `std::ifstream`.
  - Open file for reading: `open`
  - Close file after reading: `close`
  - Take input (same as `cin`, extractor operator): `>>`
- including `fstream`

16

Writing to a Text File

fit@hcmus

- using `std::ofstream`.
  - Open file for writing: `open`
  - Close file after writing: `close`
  - Take input (same as `cout`, insertion operator): `<<`
- including `fstream`

17

Examples

fit@hcmus

```cpp
#include <iostream>
#include <fstream>

int main()
{
	std::ifstream fIn;

	fIn.open("Data01.txt");
	if (fIn.is_open() == false)
	{
		std::cout << "File does not exist" << std::endl;
		return 1;
	}

	int N, i;
	int A[100];

	fIn >> N;
	for (i = 0; i < N; i++)
		fIn >> A[i];

	for (i = 0; i < N; i++)
		std::cout << A[i] << "\t";
	std::cout << "\n";

	fIn.close();
	std::cout << "Done\n";
	return 0;
}
```


18

Examples

fit@hcmus

```cpp
#include <fstream>
#include <iostream>

int main () {
   char data[100];

   // open a file in write mode.
   std::ofstream outfile;
   outfile.open("afile.txt");

   std::cout << "Writing to the file" << std::endl;
   std::cout << "Enter your name: ";
   std::cin.getline(data, 100);

   // write inputted data into the file.
   outfile << data << std::endl;

   int age;

   std::cout << "Enter your age: ";
   std::cin >> age;

   // again write inputted data into the file.
   outfile << age << std::endl;

   // close the opened file.
   outfile.close();
```


19

fit@hcmus

Questions and Answers