Handout – Basics

Editor

How to solve a task using our VSCode plugin:

- Download description and samples from the grader.
- Install VSCode and our plugin SOICode.
- "New Empty SOI Task" (in the sidebar or CTRL+MAJ+P).
- Code your solution.
- Add some testdata: Right-click→"IORun Add IO".
- Run your program using CTRL+Enter.

Basic template

```
#include <iostream>  // can replace all inclusions
#include <algorithm> // with one
#include <vector>    // #include <bits/stdc++.h>
#include <numeric>   // when not on mac
#include <tuple>
#include <utility>
// use standard library without "std::"
using namespace std;

#define int int64_t // use 64-bits ints

// main function, is always executed first
// put your code inside
signed main() {
        // input/output optimization
        ios::sync_with_stdio(false);
        cin.tie(0);
        // your code here...
}
```

Instructions

Instructions always end with a semicolon and are executed from top to bottom. Here are some useful instructions:

- Declare variable x of type 't': `t x;`
- You can declare several variables at the same time and assign them a value directly: `int a=10, b;`
- Assign a new value to an already declared variable: `a = 15;`
- You can't use the same name for two variables (more on this in a later lecture).

Some types

- **`int`**: signed integers (can be negative). Use `#define int int64_t` to use 64-bit integers or use **`long long`**.
- **`unsigned long long`**: 64-bit positive integers (when normal **`int64_t`** or **`long long`** are not enough).
- **`double`**: Floating point numbers.
- **`bool`**: Boolean truth values: `true` or `false`.
- **`char`**: Characters. Use single inverted commas, for example `char c = 'a';`
- `string`: Strings of characters. Use quotation marks, for example `string s = "Hello, world";`

Input and output

Get the input using `cin`:

```
int a, b;
cin >> a >> b;
```

Use `cout` in order to print to the output. Printing `'\n'` creates a new line.

```
int a = 1, b = 2;
cout << "a+b: " << a+b << '\n'; // Prints "a+b: 3".
```

Operations on integers

Standard operations:

- `a+b`: addition.
- `a-b`: substraction.
- `a*b`: multiplication.
- `a/b`: integer division (e. g. `5/3=1`).
- `a%b`: rest of the integer division of a by b (p. ex. `5%3=2`). Also known as 'modulo'.

Pay attention to operator precedence. Multiplications, divisions and modulos are executed before additions and substractions. Operations with the same precedence are executed from left to right.
Use parentheses to force another order of operations: `(a+b)*c-d/(e%(f*a))`.

Comments

Make your code more readable (also when you're the only one reading it) by using comments, starting by '//' and ignored by the compiler:

```
cout << (h2-h1)/(x2-x1); // outputs the slope
```

Conditionals

You can use conditional expressions to control what your program does according to the truth value of some expression. In order to do that, use the **`if`** structure.
Create a block of code between brackets `{ ... }` in order to determine which instructions are executed when the condition is met.

```
if(condition) {
        // executed if condition is true
        ...
}
```

Use **`else`** to specify what to do when the condition is not true:

```
if(condition) {
        // executed if condition is true
        ...
} else {
        // executed if condition is false
        ...
}
```

Conditions should be expressions returning a boolean value (`true` or `false`). Here are some useful expressions that you may use to test some conditions on two numbers a and b:

- `a==b` is `true` if a and b are equal and `false` otherwise.
- `a!=b` is `true` if a and b are not equal and `false` otherwise.
- `a<b` is `true` if a is less than b and `false` otherwise.
- `a>b` is `true` if a is greater than b and `false` otherwise.
- `a<=b` is `true` if a is less or equal to b and `false` otherwise.
- `a>=b` is `true` if a is greater or equal to b and `false` otherwise.

You can combine boolean expressions using logical operators:

- `!a` is `true` if a is `false` and `false` otherwise.
- `a&&b` is `true` if both a and b are `true` and `false` otherwise.
- `a||b` is `true` if either a or b is `true` and is `false` if both are `false`.

Order of operations: `!` > number operators > `==`, `!=` > `||` > `&&`.
You can nest conditionals:

```
if(x==2) {
        ...
        if(!a-3==5) {
                ...
        }
        ...
}
```

More

To learn how to do more things in C++, follow the next lectures, ask us and never hesitate to search on reference sites such as https://en.cppreference.com/w/.