C++プログラミング講習

main 関数,出力,四則演算

0. はじめに

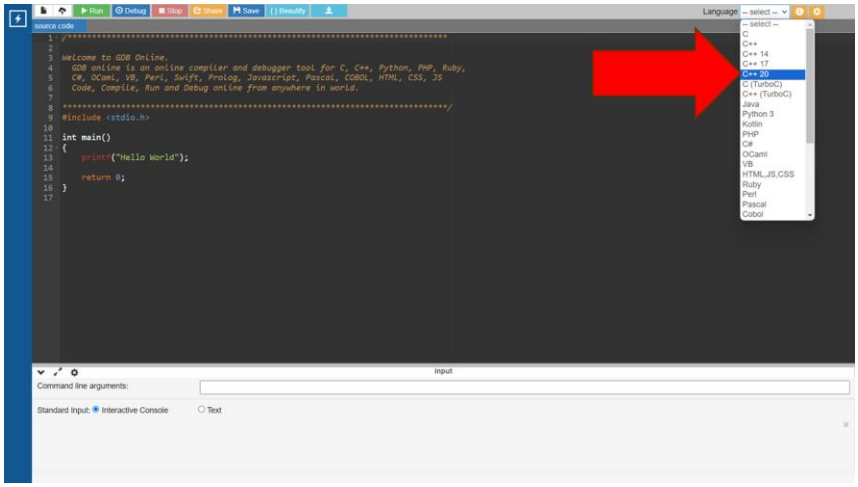
　みなさんこんにちは、高校1年生の李です。新入部員向けに C++の講習を開くことになりました。この講習では、プログラミング言語は C++を扱いますが、プログラミングの基礎の考え方を学ぶことを目的にしています。

※エディター

　ブラウザからコードを実行できる onlinegdb や、

　自分のパソコンを持っている人は Visual Studio Community 等を使用してください。

onlinegdb を使う人は、C++20 を選択してください。



1. Hello, world!

　早速ですが、コードを書いてみましょう。

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```

このコードを実行して、出力がこう表示されたら、成功です。おめでとうございます

```
Hello, world!
```

このコードの意味を早速1行目から解説して…ということをすると難しくなってしまうので一旦置いておくことにします。

2. プログラムの基本形

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5      //ここの中プログラムを書く
6      return 0;
7  }
```

　C++でプログラムを実行する時、int main()の後にある{　}に囲まれた部分に書かれているプログラムが上から順番に実行されます。

ここの部分4行目～7行目の部分をmain関数と呼びます。

これからのプログラムでは、**基本的にmain関数内だけを考えれば大丈夫**です。

また、return 0; は、main関数を終了させる命令です。省略することもできます。

//ここの中プログラムを書く　となっている行はコメントと呼び、プログラムが何をしているかを人間がメモするときなどに使います。プログラムの実行結果には影響を与えません。

あらためて、さっきの Hello, world! と出力するプログラムを見てみましょう

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5      cout << "Hello, world!" << endl;
6      return 0;
7  }
```

main関数の中の

```
1  cout << "Hello, world!" << endl;
2  return 0;
```

が実行されます。

3. セミコロン

　C++では、たいてい全ての文の後ろに ;（セミコロン）が必要です。

忘れがちなので、気をつけましょう。

4. cout

C++で文字や数字を出力するときは、cout(しーあうと)を使います。使い方は、

```
1  cout << "テキスト見本";
```

のように、まず、coutを書いた後、<<”出力したいテキスト” と書きます。

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5      cout << "Hello";
6      cout << "世界";
7  }
```

このプログラムの実行結果はどうなるでしょうか？

1.

```
Hello 世界
```

2.

```
Hello
世界
```

3.

```
Hello  世界
```

4.

```
世界 Hello
```

答えは、自分で実行して確認してみてください。

念の為に [ ]←コピーして他の場所に貼り付けると見えます。

cout は << を複数つなげることもできます。例えば上のプログラムは、

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5      cout << "Hello" << "世界";
6  }
```

と書き直す事もできます。

改行するときは、endlを使います。

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    cout << "Hello" << endl << "世界";
}
```

これの実行結果は、2番と同じになります。

```
Hello
世界
```

5. 全角文字

C++では、プログラムの中に全角文字や全角スペースがあるとエラーができます。

特に、全角スペース「　」は発見するのが難しく、よく起こるので気をつけましょう。

なお" "の中と、//コメントの中では全角文字を使っても問題ありません。

6. インデント・スペース・改行

インデントやスペース、改行は人間がプログラムを読みやすくするために使います。

```
#include <bits/stdc++.h>
using namespace std;
int main(){
cout<<"Hello";cout<<endl;
}
```

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    cout << "Hello";
    cout << endl;
}
```

上の2つのプログラムは同じことをしていますが、上は詰め込んでいて人間が読みづらいです。

インデントは通常行の最初に置きます。キーボードのTabキーを押すとできます。

7. 四則演算と優先順位

演算子	意味
+	足す
-	引き算
*	掛け算
/	わり算
%	割ったあまり

※割り算/は整数同士の場合、割って小数点以下を切り捨てした結果になるので注意

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   int main() {
4       cout << 1 + 1 << endl; // 2
5       cout << 3 - 4 << endl; // -1
6       cout << 2 * 3 << endl; // 6
7       cout << 7 / 3 << endl; // 2
8       cout << 20 % 3 << endl; // 2
9       cout << 10 + 5 * 2 << endl; // 20
10      cout << (10 + 5) * 2 << endl; // 30
11  }
```

現実世界の算数と同じように、掛け算とわり算は足し算引き算より先に計算されます。

また、カッコ()でくくることで計算の順序を変えられます。

優先順位が高い	優先順位が低い
* / % ( )	+ -

※注意点

　例えば、1から50までの数の総和を求めるとき、

```
(1 + 50) / 2 * 50
```

と書くと、まず `(1 + 50)` が実行され、`51 / 2 * 50` となります。次に実行される、`51 / 2` は整数同士のわり算なので、小数点以下が切り捨てられてしまい、正しい結果が得られません。

```
 (1 + 50) * 50 / 2
```

このようにすることで、正しい結果になります。

おまけ

　たぶん次の資料で説明しますが、C++で小数が入った数(実数)を扱うこともできます。
実数をコンピュータで処理扱うときは、近似値で扱うのが一般的です。
例えば1/3は、0.33333……と小数点以下の数字が無限に続く無限小数ですが、コンピュータでは有限の値で表す必要があります。
分子と分母を別々に持つ方法もありますが、分数同士の計算で分母が非常に大きくなるし、平方根を求められないのであまり使われないらしいです。

　C++での負の整数のあまりを求めるとき、現実の数学と違う挙動をします
例えば-5を3で割ったあまりは、数学では1になりますが、C++では-2になります。
つまり、割られる数とあまりの正負が一致します。

`#include <bits/stdc++.h>`は何をしてますか

　ソースコードが書かれているファイルをソースファイルといいます。
様々な機能を持つ関数や構造体を書いたいソースファイルをまとめたものをライブラリといいます

`#include <ファイル名>`は、includeディレクティブと言い、指定したファイルの内容をここに展開する命令です

`bits/stdc++.h` には、C++の標準ライブラリ(STL)が全部入っています

実は、coutやendlもSTLの機能です

`using namespace std;`は何をしていますか

　`namespace`は名前空間です。

名前空間は、関数や構造体、グローバル変数の衝突を避けるために定義されます。
名前空間名::関数や構造体　みたいに使います。
そして、STLの関数や構造体には、stdという名前空間が定義されています。
なので、本来STLの機能を使うには、std::を前に置く必要があります。
`using namespace std;`を書くことで、std::を省略できるようにします。