競プロ講習会 (初級)c++ 編

KARASU4280

2020 年 6 月 20 日

目次

1　自己紹介

名前:KARASU4280
UEC19 2 年 類 I3 クラス

プログラムは大学に入ってから勉強しました。
入学後半年かけて C 言語を勉強したが、c++ の方が楽だと感じて c++ に移行した。
Atcoder の ABC163 がコンテスト初参加！しかし、Unrated...
現在は灰色
今年中に緑になれるように頑張ります。

2　C++

c++ は覚えることは多いですが、その分出来ることも多く、計算も速いことが特徴の言語です。

2.1　Hello,World

ほとんどのプログラミング言語には、形があります。
ここでは、c++ の形を以下のプログラムで説明します。
Hello,World のソースコードを、1 に示す。

ソースコード 1　Hello

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

順に解説していきます。

1 行目 ヘッダファイルを有効にする

c ++ のヘッダファイルには、標準ライブラリ関数が宣言されている。
これらを有効にしないと標準ライブラリ関数が使えないので、include の形で有効にする。
bits/stdc++.h というヘッダファイルは、たくさんあるヘッダファイルを全て有効に出来るので、これを覚えておけば大丈夫。

2 行目 名前空間の概念

c++ には、名前空間という概念があります。

名前空間とは、ヘッダファイルに書かれている関数の名前が入っている所です。
ヘッダファイルの関数を使うとき、○○という関数を使うとだけ言っても使用できません。ちゃんと××という名前空間にある○○という関数を使うと言わないとだめです。
しかし、いちいちヘッダファイル内にある関数を使うときに××という名前空間にある○○という関数を使う、と入力するのは面倒です。
そこで 2 行目のような書き方をして、○○という関数名が出てきたらもうそれは××にある○○という関数のことだ、とコンパイラに伝えます。

また、c++ では一行分の処理が終わったら; が必要です。

4 行目 main 関数の宣言

c++ でプログラミングをするとき、好き勝手関数に名前をつけられますが、どの関数から処理したら良いのかわからなくなります。なので、一番はじめに処理するという目印が必要になり、それが main 関数です。
この関数に書かれている事が一番はじめに実行されます。
関数の中身は {} で囲みます。

5 行目 文字列の出力

あとでまた説明します。
cout はヘッダファイルに書かれている関数で、コンソール画面を意味するものです。それに、”<<” という出力演算子を使って Hello,World を画面上に出力します。

\n は改行を意味する文字です。これがあるところで改行します。
そして、1 行の終わりに; を入れています。

6 行目 正常終了

main 関数は終了するときに呼び出し側のシステムに対して整数値を返す機能を持っています。
そのとき、return 整数; と書き。正常終了なら 0、問題があって終了するときは 1 とするのが慣例です。
省略しても問題が無い場合が多いですが、Atcoder のコンテストに限っては、この分を忘れるとジャッジされないので必ず必要です。

以上 cout の行を除いたものが c++ の基本的な形になります。
これから、プログラムを書くための説明に移ります。

3　入力、出力、浮動小数点 (誤差は含まない)

3.1　型

　プログラムを書くとき、
c++ は、厳密に型指定された言語であり、静的に型指定されます。全てのオブジェクトには型があり、その型は変更されません。
以下の表は、基本的な型の一部です。

名前	サイズ	役割
int	4 バイト	整数値の既定オプション
char	1 バイト	文字の既定オプション
string	1 バイト	char 型を使いやすくした文字列の型
double	8 バイト	浮動小数点値の既定オプション
void	0 バイト	変数の宣言には使えない。型を宣言しない特殊な型
long long	8 バイト	int に入りきらない整数を表す型
float	4 バイト	浮動小数点値を表すが double 型より精度が悪い

　変数を宣言するとき、これらの型と変数の名前を同時に宣言する。
型と異なったデータを変数に渡すと予期せぬ挙動をする事がある。

3.2　入力

　変数を宣言した後、cin という関数で 2 のようにその変数に入力できる。

ソースコード 2　cin

```
1 int a;
2 string b;
3 cin >> a >> b;
```

変数を宣言した後、”>>”という入力演算子を使って宣言した変数に入力している。

3.3　出力

　出力には、形の説明時に使った cout を使う。
cin のプログラムの変数を出力 3 で出力してみる。

ソースコード 3　cout

```
1 int a;
2 string b;
3 cin >> a >> b;
4 cout << a << ' '<< b << '\n';
```

となる。

4　代入、四則演算

4.1　代入

変数にデータを格納することを代入と言い、変数の宣言時にそれをすることを変数の初期化と言います。
代入の仕方は 4 で示します。

ソースコード 4　assign

```
1 int a = 0,b;
2 double p = 3.14;
3 a = a + 6;
4 b = (int)p;
5 cout << a << ' ' << b << '\n';
```

このように、”=”を用いて表現します。
1 行目では、変数 a を 0 で初期化しています。すると、a の中身は 0 となります。b のように初期化をしないことも出来ます。その場合は適当な値が格納されます。
3,4 行目で代入しています。ここで使用している”=”は左辺に右辺を代入するという意味です。
しかし、型の異なるデータを代入することは原則出来ません。そうするためには、キャストと呼ばれる型変換の操作が必要です。
4 行目は double 型のデータを int 型にキャストして代入しています。
キャストの仕方はいろいろあるのでこの書き方が分かりづらい人は調べてみてください。

4.2 四則演算

四則演算の時に使う演算子を以下の表で説明します。

演算子	操作	備考
+	加算	
-	減算	マイナス記号
*	乗算	
/	除算	int 型同士での除算は小数点以下は切り捨てられる。たとえば、10/3 は 3 となる。
%	剰余算	ともに整数の場合余りを計算する。たとえば、10%3 は 1 となる。
++	インクリメント	1 増やす。
–	デクリメント	1 減らす。

これを実際にソースコード 5 で確認する

ソースコード 5　calc

```
1 int a = 5,b = 4,c;
2
3 c = a + b;
4 c = a - b;
5 c = a * b;
6 c = a / b;
7 c = a % b;
8 a++;
9 a--;
```

計算結果は、上から順に 9,1,20,1,1,6,5 となります。

5　問 1

　この問題を考えてみましょう。

https://atcoder.jp/contests/practice/tasks/practice_1

この問題は、今まで説明した範囲の知識で解くことが出来ます。

ソースコード 6　ans1

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

int main (void){
  int a,b,c;
  string s;

  cin >> a >> b >> c >> s;
  a = a + b + c;
  cout << a << " " << s << '\n';
  return 0;
}
```

問題文にあるように、変数の宣言をします。そして、a+b+c を計算して、文字列と一緒に出力します。

6　二方向分岐・文字列操作

6.1　二方向分岐

もし〜ならば〜という形式で二方向分岐するプログラムを書くときには if 文を使います。

イメージ

もし　〜　なら
これをする。
そうでなければ
これをやれ

というものになります。

書式

if(条件式){
文 1
}else{
文 2
}

という書き方になります。
条件式が真なら文 1 を、偽なら文 2 を実行します。
else 以下は不要なら省略できます。
単文なら {} を省略出来ます。

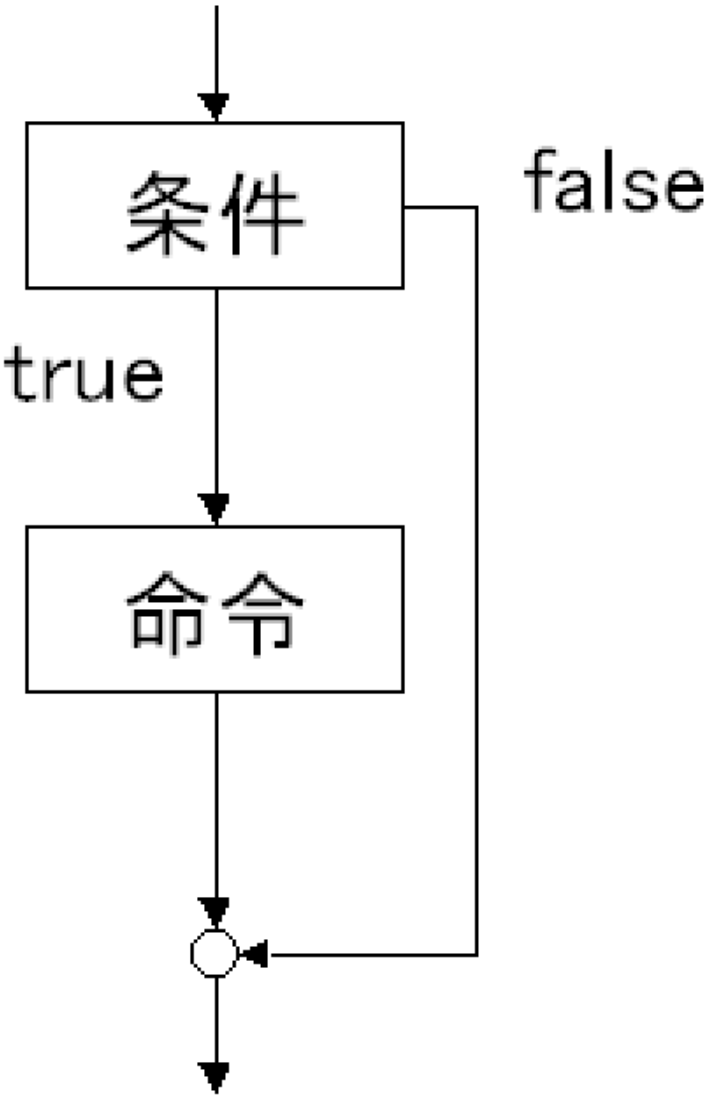

図１　分岐のイメージ

ここで、条件式を説明します。
条件式は以下の表にある関係演算子というものを用いて記述します。

演算子	操作	備考
>	大きい	
>=	以上	
<	未満	
<=	以下	
==	等しい	代入と間違えやすいので注意
!=	等しくない	!には否定の意味がある。
&&	かつ	条件を二つ書く時に用いる。
||	または	条件を二つ書く時に用いる。

先ほど、単文の時はを省略可能と説明しましたが、その例を示します。
単文の例

ソースコード 7　one

```
1  if(b > 10)
2    b = 20;
3  else
4    b = 50;
```

文 1, 文 2 が複文ならが必要です。

ソースコード 8　more

```
1  if(b > 10){
2    b = 20;
3    a = 10;
4  }else{
5    b = 50;
6    a = 30;
7  }
```

また、if 文は基本二方向分岐するものですが、else if 文を続けることでいくつでも分岐させることが出来ます。

ソースコード 9　else if

```
1  if(b > 10){
2    b = 20;
3    a = 10;
4  }else if(b > 20){
5    b = 30;
6    a = 20;
7  }else if(b > 30){
8    b = 40;
9    a = 30;
```

```
10  }
```

6.2 文字列操作

char 型は本来は文字型なので、文字列を扱う場合は、char 型の変数をたくさん作り、つなげる事で対応してきた。
しかし、いろいろな問題が生じた。
その問題を解決するために作られたのが string 型。
なので、string 型の方が文字列操作がわかりやすいので、string 型を用いて文字列操作を説明します。
string 型は可変長の文字列処理をする型なので、宣言するときに文字数を指定する必要は無いです。

string 型の初期化 string 型はいろいろなスタイルで初期化できます。

ソースコード 10　string

```
1  string s1;
2  string s2 = "";
3  string s3 = s2;
4  string s4(4,'Z');
5  string s5 = "abcd";
6  string s6(s5,1,2);
```

これら全て string 型を初期化する方法です。
順に、s1,s2 は空文字、s3 は s2 と同じ文字列、s4 は Z4 文字、s5 は abcd,s6 は s5 の 1 文字目から 2 文字という風に初期化されます。

string 型対応の演算子以下の表は string 型を扱える演算子である。

これらを使用するに当たって注意する点があります。

演算子	操作	備考
=	代入	
+	連結	二つの文字列を連結する
+=	連結代入	右辺の値を左辺の文字列の末端に追加する。
==	等号	
!=	不等号	
<, <=, >, >=	比較	
<<	出力	
>>	入力	

それは、string 型ではない文字列を操作する場合です。
演算は左から右に進むので、演算の先頭は string 型である必要があります。

ソースコード 11　caution

```
1  string s1,s2;
2  s1 = "aaa" + "bbb";
3  s1 = "aaa" + "bbb" + s2;
4  s1 = s2 + "aaa" + "bbb";
5  s1 = s2 + "aaa" + "bbb" + "ccc";
```

上記のコードのうち、上から 2 行はエラーです。
演算の先頭が string 型であれば、演算結果は string 型になりますが、そうでない場合、演算子が対応していないのでエラーになります。

特定の文字を操作する。

添字演算子 [] を用いて文字列中の特定の文字を操作できる。

ソースコード 12　str

```
1  string s1 = "abcde";
2  s1[2] = 'Z'
```

string 型は先頭の文字を 0 番目として数えるので、上記の s1 は abZde となります。
このように、文字列の任意の文字が取り出せます。

メンバ関数

string 型オブジェクトはいくつかの機能を持っています。
これらは string クラスのメンバ関数と呼ばれています。全部説明するのは重いので最低限知ってほしい 2 つを紹介します。

ソースコード 13　member

```
1  string s1 = "abcde";
2  cout << s1.at(2) << '\n';
3  cout << s1.size() << '\n';
```

2 行目の s1.at(2) は s1[2] と同じ意味です。しかし、文字列の有効範囲を指定すると、スルーしてくれます。s1[] で有効範囲外をしているとエラーになります。
3 行目の s1.size() は文字列の長さを返してくれます。
他にも便利で比較的よく使うメンバ関数もあるので、不自由だと感じたら調べてみてください。

7　問 2

　この問題を考えてみましょう。

https://atcoder.jp/contests/abc093/tasks/abc093_a

ソースコード 14　ans2

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

int main (void){
  string s;
  cin >> s;
  if(s[0] != s[1] && s[1] != s[2] && s[2] != s[0]){
    cout << "Yes\n";
  }else{
    cout << "No\n";
  }
  return 0;
}
```

'a','b','c' からなる 3 文字の文字列を並び替えて"abc"になるためには、3 文字全て異なっていれば良い。

8　指定回数繰り返し

指定回数同じ処理を繰り返すには、for 文を使います。

書式

for(初期化式; 継続条件式; 再初期化式){
文
}

c++ の for 文では、ループに入る前に初期化式を実行しする。その後、継続条件式が真の間文を実行する。各ループ処理のたびに文の実行後、再初期化式を実行する。
という流れになってます。
継続条件式の書き方初期値は、通常の代入文を書きます。継続条件式はこの式が真の間ループを続けるという意味です。
指定回数反復処理する文は 15 の様になります。

ソースコード 15　for

```
for(int i = 0;i < 10;i++){
  a = a + i;
}
```

この文は for の中の文を 10 回繰り返すという意味です。
i = 0 という初期値を設定し、i ¡ 10 が真の間ループのたびに i を 1 ずつ増やしてループします。
この i は for 文の中で変数として使用できます。

9 問 3

この問題を考えてみましょう。https://atcoder.jp/contests/abc093/tasks/abc093_b

ソースコード 16　ans3

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

int main (void){
  int a,b,k;
  cin >> a >> b >> k;

  if(b - a + 1 >= 2*k){
    for(int i = 0;i < k;i++){
      cout << a + i << '\n';
    }
    for(int i = 0;i < k;i++){
      cout << b - k + 1 + i << '\n';
    }
  }else if(b - a + 1 >= k){
    for(int i = 0;i < k;i++){
      cout << a + i << '\n';
    }
    for(int i = 0;i < b - a + 1 - k;i++){
      cout << a + k + i << '\n';
    }
  }else{
    for(int i = 0;i < b - a + 1;i++){
      cout << a + i << '\n';
    }
  }
  return 0;
}
```

分岐は 3 つあります。

b - a + 1 >= 2*k の時と b - a + 1 >= k の時と、b - a + 1 < k の時です。1 番目のケースは、小さい方から k 番目の数と大きい方から k 番目の数がかぶりません。なので、素直に実装します。

2 番目のケースは、小さい方から k 番目の数と大きい方から k 番目の数がかぶります。なので、かぶっている部分をループから外すために、出力を a + k の次から最後までループさせます。

3 番目のケースは、小さい方から k 番目の時点で b まで出力されるケースです。この場合は、a 以上 b 以下の整数を全て出力すれば良いです。

おわりに

自分に合ったプログラミング言語に出会うことで、プログラムを書く負担をとても軽減出来ます。
なので、まだ初心者のうちは 2,3 言語に触れて自分に合ったものを見つけた方が良いと思います。
ご静聴ありがとうございます。