0から始める
競技プログラミング

@rhyska(村上涼太)

はじめに(読まなくていいよ)

　競技プログラミングに対してのモチベーションに関しては、興味があるけどどんなものかわからない、joiなどのコンテストで勝ちたいといったふうに、人によって異なることと思います。

　多くの人は、競技プログラミングは難しいもので、一部の天才さん（固有名詞ではない）が参加するものだと思っていることでしょう。しかし、実際には競技プログラミングのハードルはそこまで高くなく、練習さえすれば誰でも楽しむことができますし、アルゴリズムや実装力を鍛えるとともに、自身の数学力、思考力アップにも繋がります。

　このスライドを覗いてしまったそこのあなた、ぜひこの機会に競技プログラミングをはじめ、あなたの生活に彩りを持たせてはどうでしょうか。

　さて、このスライドでは、初めて競技プログラミングにふれるという方を対象にしています。プログラミングの基礎を身につけることを一番の目的としていますので、数学が苦手でも、プログラミングをしてみたいという方はぜひご覧ください。

競プロってなんですか

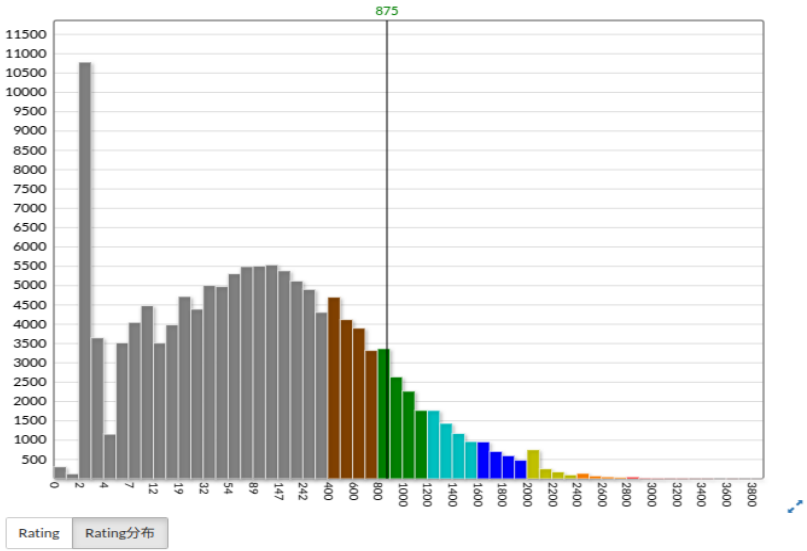
競プロisなに

競技プログラミングは、数学っぽい問題をプログラミングで解くことを競技化したものです。コンテストの流れとしては、参加者に複数の問題が与えられ、それを制限時間内にどのくらい解くことができるかを競うというものになっています。
また。問題はただ解ければ良いというわけではなく、プログラムを実行するための時間制限もあるので、*効率的な手法を考えることが求められてきます。競プロの楽しみは、いかに計算回数を少なくして問題を解くか考えることにあると言っても過言ではないです。

*例えば、もし正確な答えを出せる解法を考えられたとしても、その答えがわかるまでの時間が100万年ほどかかるというのなら、実質的には答えを出せていないのと変わりません。

AtCoderに登録しよう。

競プロをするにあたって、AtCoderというサイトを利用します。これは日本最大手のプロコンサイトで、毎週土曜日にコンテストが開催され、毎回約1万人程が参加しています。もちろん参加費は無料であり、子供から社会人まで幅広い層が参加しています。
AtCoderの利点はレートが存在することです。レートは強さの証明になりますし、競プロのモチベにも繋がります。ちなみにレートの最高位は赤で、赤色コーダーは多くの参加者に変態扱いされています。ということで、今すぐAtCoderに登録しよう。(userIDは実名非推奨です)



色は下から灰茶緑水青黄橙赤となっています。ちなみに筆者は緑色です。(2025/4/18)ちなみに、x色のレートの人のことをx色コーダーと言うことがあります。例えば、私は緑色コーダーとなります。

補足：レートについて

レーティング	色	相対的な位置	絶対的な位置
2800+	赤	上位 0.2%	
2400-2799	橙	上位 0.6%	
2000-2399	黄	上位 2%	アルゴリズムの研究職・研究開発で重宝されるレベル
1600-1999	青	上位 5%	ほとんどのIT企業でアルゴリズム能力がカンストする
1200-1599	水	上位 10%	半数以上のIT企業でアルゴリズム能力がカンストする
800-1199	緑	上位 20%	エンジニアとしてかなり優秀
400-799	茶	上位 35%	学生なら優秀
1-399	灰	上位 100%	

茶色だと学生として優秀だそうです。ということで
まずは茶色コーダーを目指してみましょう。
ちなみに橙以上は競プロガチ勢という評価です。

AtCoderJuniorLeagueについて

AtCoder社は、学生が競プロで競う場を提供する、教師に競プロを頑張っている生徒がいることを伝えるために、AJLという学生のみのコンテストを開催しています。学校入賞すればかわいいステッカーをもらえますので、ぜひ登録して、コンテストに挑みましょう。ちなみに、舞鶴高校は人海戦術でちょっといい位置にいます。OUGHの生徒はみんなで実力を上げて、OMZHをぼこしていきましょう。

順位	学校名	都道府県	スコア	参加者
1	開成高等学校	東京	380084	zeta7532 Yoyoyo8128 tkms watasou1543 kuroinusan tee_73009
2	東大寺学園高等学校	奈良	366873	Iroha_3856 sortA tikuwa_ champignondayo sunset1231 OYU__0YU
3	神奈川大学附属高等学校	神奈川	243147	hirayuu_At
4	筑波大学附属駒場高等学校	東京	221579	shiomusubi496 Magentor Falcon_ joi_ganbaru tfl_tkpc
5	茨城県立並木中等教育学校	茨城	196782	nouka28 AwashAmityOak rotti Untitle Superkikki Facade soutan159
6	福島工業高等専門学校	福島	102390	hiro1729 cno fuvxx
7	筑波大学附属高等学校	東京	91792	raspberry1729 Astroseek yaakiyu taiki1121 Nom8 Orange_pekoe neko10 rincotan miyamonn
8	広島大学附属高等学校	広島	67963	po1 RC0 Shymohn
9	西南学院高等学校	福岡	64076	VIP1109
10	京都府立南陽高等学校	京都	62862	katsuta Saki1103 Art_118 hisui527 sapphire7110
11	早稲田高等学校	東京	54385	yama_can tomo8 Houjitya_K
12	静岡県立静岡高等学校	静岡	50950	leon1010 chihi311 hirosoro ferrite
13	大牟田高等学校	福岡	50505	kusuninu
14	明石工業高等専門学校	兵庫	48806	bandai0412 NandK imak Meshel Pon12345
15	大分県立大分舞鶴高等学校	大分	48127	pcsuke azumak makoron_6 TamaiKoyomi launchpencil soom satouhao niko0906 AORNG kouki0404 tomatyu tori101500 kazurei rukaa mikiya1203 taisei tkiyom
16	帝塚山高等学校	奈良	43785	rurumaru ta2maru yuhidon haku89
17	徳島県立城ノ内中等教育学校	徳島	41074	potat_p riu0415 HeyHeyHello Jsoreike29 bem130 S0ftGummy
18	S高等学校	茨城	35267	mamahi tacopic007 athfhf

おべんきょの仕方について

AtCoderに登録したことと思いますので、早速勉強を初めていきましょう。このスライドでは、AtCoderのc++学習支援教材であるAPG4bを使っていきます。アクセスすると、参加登録のボタンがあるはずですので、押しておきましょう。一定の学力を持っている方であれば、APG4bを読むだけでだいたいを理解できると思いますが、いくつか注意点がありますので、気をつけましょう。
・「キーポイント」は読み飛ばして良い
どうせ後に詳しく書いてるので、わからなかったら飛ばしましょう。復習する際には便利ですよ
・「詳しい話」はできればちゃんと読もう
数学の難しい話などがあれば飛ばして良いですが、基本的には重要事項が書かれていることが多いですので、しっかり読むことを勧めます。
・ページの下の方にある問題は頑張ってとこう
「難しい」は理解できなければ飛ばして良いです。
・コードの実行は「コードテスト」を使おう
もし環境構築したい場合は、本スライドの著者に連絡しよう。

本スライドでは、つまずきやすいポイントを解説していきます。困ったときは、是非活用してください。また、どうしてもわからない場合は友達に聞いたりdmしたりしてください。

コーディングについて

競プロでは、早くコーディングすることが求められ、ソースコードを後でじっくり見返すという行為もあまりしません。そのため、世間でよく言われる「見やすいコードを書け」は、ある程度無視しても構いません。例えば、自分が読みやすい！と思わない限り、無駄な空白を入れるのはやめましょう。APG4bで空白が入れられていても、実は入れなくてもいいという例はたくさんあります。頻繁にスペースキーを押すことはストレスになるので、
良ければ無駄なスペースはなくすことを勧めます。

1.00.はじめに

提出後にWAやCEが出る場合は、「c++23(gcc12.2)」が選択されているか確認してください。

1.01.output and comment

使うことはないですし、使うことは推奨しません（むしろ大迷惑）が、コメントを打つ際に、末尾に\をいれるとその下の行もコメントアウトされます。たまにバグの原因になるので、気をつけましょう。（一部の環境では”能”という文字で下の行に影響が出ることがあります。）
コードの先頭に書くおまじないについての質問があれば、友達に聞きましょう。

* 細かいはなし
出力する際に、cout<<(文字列)<<endl;としました。endlは改行するためのものですが、同時にflush(理解する必要はない)という動作をするため、実行時間が長いです。よって、改行したいときは改行文字として,’\n’という文字がありますので(後の単元で学習します)、そちらを使うことをおすすめします。
例えば、cout<<(文字列)<<”\n”;というふうになります。

1.02.書き方とエラー

結構どうでもいい話ですが、c++において、ホワイトスペースはすべて区別されません。
つまり、spaceもtabも\nもすべて一緒なのです。基本的に、コードは横に長くなると読みにくくなる傾向にあるので、適宜スペースのかわりに改行するなどすると、コードを読みやすくなることがあります。

1.03.演算

特に言うことはありませんが、演算子の優先順位は引っかかりやすいところなので、演算子の仕様をちゃんとおべんきょして優先順位を理解したり、自信がない場合は括弧で囲ったりしましょう。

1.04.変数と型

数字系の型には、intやdoubleなどがあります。コンピュータでは当然無限桁を表すことはできないので、例えば小数(実数)を使いたいときにはdouble(浮動小数点数)を使います。浮動小数点数が何かは各自調べるものとして、doubleは有限桁での実装なので、誤差が出ることがあります。皆さんご存知の通りコンピュータは2進数を用いて実装されているので我々が普段使っているような有限小数でも、彼らにとっては循環小数となる場合があるのです。
ちなみに、intはだいたい2^30くらいまでの数を表すことができますが、それ以外にもlong long(2^60くらい)などの型もあります。
文字列はstring型で表されましたが、文字はchar型で表します。覚えておきましょう。stringは”で、charは’で囲みます。

1.05.入力

ここでは、難しめの問題をどう解くかを解説します。

Round up the mean・・・(a+b)÷2を切り上げて。
まず、切り捨ての割り算で、a÷bを切り上げる方法を考えてみましょう。切り上げなので、aがbで割り切れるときはa/b,その他のときはa/b+1となります。この時、(a+b-1)/bとすると、切り上げが達成できることがわかります。これは、aをbで割ったあまりに注目するとわかるので、ぜひ考えてみてください。
よって、この問題では(a+b+1)/2を出力すれば良いです。

Placing marbles・・・1は何個ありますか。
x00のxが0か1か判定することを考えましょう。0のとき、もちろん値に変化はありません。1のときは、xが何桁目にあるかに関わらず、必ず9*(桁数-1)+1となることがわかります。つまり、xが1である時、入力を9で割った余りが1増えるのです。よって、この問題では9で割った余りと1の数は等しく、n%9を出力すれば良いです。

1.06.if文

(難し目の内容なので、飛ばしても構わない)読みすすめていて、疑問を感じませんでしたか？if, elseときて、elseifが来るかと思いきや、else ifが来ましたからね。それもそのはず。なんと、c++にelseif文はありません。あるのはifとelseだけです。それではどうしてelse ifを表現できるのかという話ですが、それにはc++の仕様が大きく関わっています。それは「ぶら下がりif/else文」というものです(勝手に名付けました)。より詳しく説明すると、c++では、if()やelseの後にif()を書くと、次のelseはその書いたifに対応するというルールがあります。
このルールを踏まえてelse ifを見てましょう。
①if(){}
②else ③if(){}
④else{}
もうわかりますね。まず①と②のifelseが対応しています。そして、②のelseについて、③または④という分岐をしているのです。ですので、機能に忠実にインデントすると、次のようになります。

```
①if(){}
②else{
        ③if(){}
        ④else{}
}
```

1.07.bool

前の章で、if(x=10)が常に実行されるとありましたが、その仕組みを軽く説明すると、まず、説明にあったとおりifの中身はtrue/falseが判別できるならなんでもいいです。では、x=10はどんな値なのかというと、「10」になっています。まだ説明されていませんが、c++でx=yという式は、xにyを代入して、そのままyという値を返しています。(関数の単元で学習します)したがって、例えばif(x=0)というif文は実行されません。

1.08.ブロックとスコープ

ブロックに関して、if文でなくても{}で囲むとブロックを作ることができます。変数をある範囲でだけ使いたいというときは、ブロックを有効活用しましょう。
また、機能別に分けやすく、見やすくなるという利点もあります。

1.09.複合演算代入子

x+=yやx*=yはyを返します。また、x++やx—はxを、++xや—xはx+1やx-1を返します。
よって、Ex.9は次のように短く書くことができます。

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 int main(){
4   int x, a, b;
5   cin>> x >> a >> b;
6   cout<< ++x << "\n" << (x*=(a+b)) << "\n" << (x*=x) << "\n" << --x << "\n";
7 }
```

1.10.while

while文という名前の通り、条件式がtrueの間、処理を実行し続けます。ループ変数を0から始めるのは、色々と便利なことが多いので、慣れましょう。

1.11.for文

　人によっては難しく感じる単元なので、長々と解説します。

HinaArare

4種類入っている時はYが含まれているはずです。よって、for文ですべての文字を確認して

Yがあるかどうか判定すれば良いです。

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(){
  int n;
  cin>>n;
  bool four=false;
  for(int i=0;i<n;i++){
    char c;
    cin>>c;
    if(c=='Y')four=true;
  }
  if(four)cout<<"Four\n";
  else cout<<"Three\n";
}
```

ここでは読みやすさのためにtrue/falseを使っていますが、

実際には1/0を使うことが多いです。楽なので。

Theater
同じ座席に複数人が座ることはないので、for文で入力を受け取って、答えに加算していきましょう。

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(){
  int n;cin>>n;
  int ans=0;
  for(int i=0;i<n;i++){
    int l,r;
    cin>>l>>r;
    ans+=r-l+1;
  }
  cout<<ans<<"\n";
}
```

ハーッシャッド数

for文を使って桁和を求めましょう。xが0より大きい間、xの下一桁を足して、xを10で割れば良いです。(whileを使っても構いません)

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(){
  int n;
  cin>>n;
  int fx=0;
  for(int x=n;x;x/=10)fx+=x%10;//条件式で、xはx!=0という意味。(0がfalseそれ以外がtrueなので)
  if(n%fx==0)cout<<"Yes\n";
  else cout<<"No\n";
}
```

Addition and Multiplication
最初のうちは2倍して、その後kを足していけばいいことがわかります。for文の中で場合分けしましょう

Collecting Balls (Easy Version)
for文の中で、それぞれについて、距離の絶対値を求めたり、小さい方の値を求めたりするので場合分けが少しめんどくさいかもしれません。

Palindromic Numbers
A<=x<=bの範囲でfor文を回して、それが回分数かどうか判定すれば良いです。

Shift only
各数について、2で割ることのできる回数の最小値が答えとなります。後で学ぶビット演算を使うと楽にとけます。

1.12/13.文字列と文字、配列

ｎ文字目を.at(n)で表すとありますが、[n]でも表すことができます。こちらのほうが見やすく、書きやすいので、こちらを使うことを勧めます。
Minesweeperについては、二次元のh*wの盤面を長さh*wの文字列とみなしてとくことが可能です。

1.14.stl関数

ちょっとずつで良いので、c++の標準ライブラリの機能を覚えていきましょう。

1.15.関数

Ex問題のサンプルコードは、ときに冗長で、理解しにくいことがあります。
わからないときは、検索したりAIや友達に聞いたりしましょう。

2.00

AtCoderBeginnerSelectionを解いてみることをおすすめします・

2章は飛ばし飛ばしで説明するので、わからなかったら聞いてください。
https://onlinejudge.u-aizu.ac.jp/courses/lesson/2/ITP1/1
会津大学が運営しているaojというサイトでもプログラミングの基礎を学ぶことができるので、ぜひチャレンジしてみてください。

多重ループの抜け方

4章で習うラムダ式という関数もどきを使うと、多重ループをreturnだけで抜けることができるので便利です。他には、for文でbool(0か1)を定義するのもありです。

for(int i=0,a=1;i<n&&a;i++)//一気に多重for文を抜けたい時、aを0にすれば実行されません

参照

例えば、xという3次元配列を5回位更新したいとしましょう。通常ならx[i][j][k]を5回も書かなければなりませんが、例えばauto &a=x[i][j][k]と参照を定義すれば、aを5回書くだけでいいです。

再帰関数

理解するまではかなり難しい単元です。わからなかったらすぐに友達に聞くようにしましょう。APG4bにある問題は、中級者向けですので、とけなければすぐにあきらめまるか友達に聞くかしましょう。再帰関数はまじで挫折します。まあ、再帰関数に関しては実際にはそこまで難しくなくて、APG4bの教え方に問題があるという見方もできます(高校数学の内容を使っているため)。わからなくても落ち込まず、さっさと次の学習をすすめましょう。

計算量

計算量は競プロにおいて一番大切な要素です。答えを出すのに100万年かかったら意味がないですからね。
STLの関数などの計算量を知ったときは、どのようにしてその計算量になっているか考えるとアルゴリズム力が高まると思います。

構造体

使いこなせると便利な型ですが、APG4bの説明がわかりにくいですし、使わなくてもなんとかなることが多いので、無理に理解しなくても良いです。理解したいときはぜひ友達に質問しましょう。

ビット演算

競プロでbitsetを使うことはほぼないので、別に覚えなくていいですが、ビット演算についてはしっかり学習しておきましょう、例えば、shiftonlyは次のようにとけます。

```cpp
#include<iostream>
using namespace std;
int main(){
  int n;
  cin>>n;
  int x=0;
  for(int i=0;i<n;i++){
    int a;
    cin>>a;
    x|=a;//or演算で、1が一個でも立ってたら1にする。
  }
  cout<<__builtin_ctz(x)<<"\n";
}
```

__builtin_ctzは右に何個0が並んでいるか数えるgccの組み込み関数です。
末尾0の個数は2で割れる回数と同義です。

その他の機能

条件演算子、短絡評価、ラムダ式についてはしっかり学習しておくことを勧めます。
その他はふーんくらいでいいです。
next_permutationの例題は解いておきましょう。

第4章

当然やったほうが深い学びにつながって良いのですが、競プロに必須ではないですし、混乱する内容もあると思うので、やってもやらなくてもいいです。わからなかったらすぐに友人に質問しましょう。

つぎのすてっぷ

ここからは競プロに必要なテクニックを身につけていきましょう。
おすすめの勉強法として、atcoderの過去問をときながら、解説に知らない言葉があったら、それについて詳しくしらべるというものがあります。
過去問演習にはnovistepsや、atcoder problemsなどのサイトを使ってください。
novistepsは本当に素晴らしいサービスで、わかりやすい順序になっているので、APG4bが終わったあとはnovistepsを活用することを勧めます。
もしなにかわからないことがあったら友達に聞くのが一番です。みんなで協力して、力を高めましょう。