1. インデントの話

インデント(字下げ)とは、文の行頭に適当な空白やタブを入れることである。

これを行うことで、ソースコード内での処理の位置関係が分かりやすくなる。例えば、以下のようなソースコードがあったとする。

|  |
| --- |
| #include<stdio.h>  int main(void) {  int a;  scanf("%d", &a);  if (a % 7 == 0) {  printf("7 divides %d.\n", a);  } else {  printf("7 doesn't devide %d.\n", a);  }  return 0;  } |

このコードの書き方だと、どの{}がどこの処理に入っているのかわかりにくい。そこで、字下げを行ってみると以下のようになる。

|  |
| --- |
| #include<stdio.h>  int main(void){  int a;  scanf("%d",&a);  if(a%7==0) {  printf("7 divides %d.\n",a);  }else{  printf("7 doesn't devide %d.\n",a);  }  return 0;  } |

int main()の中身の処理がどこなのか、if文の中身の処理がどこなのかが分かりやすくなり、まあまあ読みやすくなった。

1. きれいなソースコードの話

前項のコードをもう少しきれいに整形しよう。まずは、適度に改行を入れてみよう。

|  |
| --- |
| #include<stdio.h>  int main(void){  int a;    scanf("%d",&a);  if(a%7==0) {  printf("7 divides %d.\n",a);  }else{  printf("7 doesn't devide %d.\n",a);  }  return 0;  } |

・#include<stdio.h>とint main()の間に改行

・変数宣言とそれ以降の処理との間に改行

・最後のreturn 0;の前に改行

というルールで改行を入れてみたが、これは個人の美的感覚なので過度に参考にしなくてもよい。

次に、適度にスペースを入れてみよう。

|  |
| --- |
| #include <stdio.h>  int main(void) {  int a;    scanf("%d", &a);  if (a % 7 == 0) {  printf("7 divides %d.\n", a);  } else {  printf("7 doesn't devide %d.\n", a);  }  return 0;  } |

スペースを入れたのは次の部分である

・#include と <stdio.h> の間

・{ の直前

・カンマの直後

・ifの直後

・各種演算子の前後(ここでは % と ==)

・else の直前

スペースの入れ方においてもある程度個人の美的感覚による。例えば計算対象同士がつながっていることを強調したいなら、次のようにあえてスペースを空けずに書くことがある(他の人はどうかわからないが、少なくとも筆者はそうである)。

|  |
| --- |
| 2 \* (A\*B + B\*C + C\*A) /\* A, B, Cの間の\*前後にスペースを入れていない \*/ |

さて、わざわざスペースや改行を空けてソースコードをきれいにする理由とは何だろうか。

一つは、「自分が読みやすい」からである。見やすいことで、プログラムの流れが把握しやすくなり、バグの発見が容易になることがある。

二つ目は「他人が読みやすい」からである。ソースコードの読みやすさは、そのプログラムの中身の理解のしやすさにある程度比例する。直近の例を挙げるなら、1年後期の「プログラミング演習」でコードを提出することになるので、教員にとって読みやすいコードを書くのが望ましい。また将来的には、企業にて自分のコードを読まれる機会があるかもしれないので、その時のためにもきれいなコードを書くことを心掛けておくべきだろう。

もちろん、コードの美しさというのは「スペース・タブ・改行をつけることによる美しさ」だけではない。論理的な整合性、処理の簡潔さ、分かりやすい変数名、オブジェクト同士の関係など様々な要因が美しさを決定する。ただ、競技プログラミングは「速く正確に問題を解くこと」に重きを置いているため、そのような美しさについて過度に気にする必要はない。