1. Practice Aを解けるようになるために

『PracticeA - はじめてのあっとこーだー（Welcome to AtCoder）』を見てみる.  
この問題が解けるようになるには

•整数a, b, cと文字列sを入力 →入力処理,入れ物

•a + b + cを計算 → 足し算

•a + b + cとsを出力 → 出力処理

つまり, 入力する処理, 入力したものをいれるもの, 四則演算, 画面に表示する処理を学ぼう.

1. 画面に文字を表示する

まずは画面上に

|  |
| --- |
| Hello, World!  Good Bye, World. |

と表示するプログラムを書いてみよう.

|  |  |
| --- | --- |
| Sample1\_1.c |  |
| #include <stdio.h>  int main() {  printf("Hello, World!\n");  printf("Good Bye, World.\n");  return 0;  } | |

実際に「Hello, World!」という表示を行っている部分は,

printf("Hello, World!\n")

の部分だけ.

現時点では, 以下の点について理解していればよい.

* 処理はまず　int main()の後の中括弧の中から始まる.上から順に命令が実行される. 現時点では,printfの部分以外は定型文として覚えてもらってよい.
* 命令の後には必ずセミコロン 「;」 を入れる.
* printf(“〇〇〇”)と入力すれば, 〇〇〇を表示してくれる.
* printf中にある『\n』は, 改行を表す文字.
* printfを呼び出すためには, #include <stdio.h>を書かなければならない.
* return 0でプログラムが終了する.

1. 数字を間接的に表示する

37 + 24の結果を表示するプログラムを作ってみよう.直接61を表示するプログラムを作ってみてもいいが, 今回は実際にコンピュータ内部で計算させてみる.

|  |  |
| --- | --- |
| Sample1\_2.c |  |
| #include <stdio.h>  int main() {  printf("37 + 24 = %d\n", 37 + 24);  printf("%d, %d, and %d\n", 64, 3 + 4, 128);  printf("Pi is %f", 3.14);  printf("%c, %s", 'A', "Hello");  return 0;  } | |

実行結果は以下のようになる.

|  |
| --- |
| 37 + 24 = 61  64, 7, 128  3.1400  A Hello |

printf("37 + 24 = %d\n", 37 + 24);

とは, 「『37 + 24 = %d』を表示しろ.ただし, %dは37 + 24を計算した値に読み替えてくれ.」という意味.

%dというのは, 「カンマで区切られた先にある値を整数で表示せよ」という意味.

printf("%d, %d, and %d\n", 64, 3 + 4, 128);

%dを複数書けば,一つのprintf中に複数の値が表示できる.そのときは,複数のカンマで複数の値を区切る.値はその順番ごとに表示される.

printf("Pi is %f", 3.14);

printf("%c, %s", 'A', "Hello");

表示できるのは整数だけではない.%fを用いれば実数が表示される.%cで文字,%sで文字列を表示することができる.

[補足]

文字は「一文字」を表し, これを表現するためにはシングルクォーテーションでくくる.文字列は「文字の集まり」を表し, これを表現するためにはダブルクォーテーションでくくる.

1. 変数の宣言と代入

プログラミングの世界には「変数」という言葉がある.

数学で習った「変数」とは微妙に意味が違う.

プログラミングにおける変数とは, 簡単に言うと

何か値を入れておくための箱

である.箱の中に必要なものを入れておけば, それを後で計算に用いたり, 書き換えたりできる.

箱には(基本的には)何でも入れていいわけではない.入れるものに適した箱が必要である.また, 箱に入れられるものの大きさには限度がある.

変数においても同じことが言える.箱の種類のことを「型」と呼ぶ.変数には整数型や文字型, 実数(浮動小数点数)型などの変数がある.それぞれの箱には決まった種類の値しか(基本的には)入れられない.また, 型によって入れられる値の大きさが決まっている.ただし, 変数一つにつき一つの値しか入れられないことに注意.

それを踏まえて, 変数を使ったプログラムのコードを見てみよう.

|  |  |
| --- | --- |
| Sample1\_3.c |  |
| #include <stdio.h>  int main() {  int val1, val2;  val1 = 10;  val2 = val1 + 35;  printf("val1 is %d\n", val1);  printf("val2 is %d\n", val2);  return 0;  } | |

実行すると以下のような結果となる.

|  |
| --- |
| val1 is 10  val2 is 45 |

{}内の処理について順に説明する.

int val1, val2;

これは「int(整数)型の変数val1とval2を宣言する」ということを意味している.

変数を作る処理を「変数の宣言」と呼び, 次の書式で書く.

型 変数名;

型には様々な種類があるが, 今のところは

int, char, double

の三つを知っておけばよい.それぞれ整数, 文字, 実数を表す.

変数名は, 「他の変数と名前が重複しない」, 「予約語(C言語が持っている特別な単語)でない」限りは, どんな名前でもよい.

カンマで区切ると, 同じ型の変数を複数宣言できる.

次に,

val1 = 10;

によって, 「val1に10を代入する」という処理を行う.

変数名 = 値;

によって, 作った変数の中に値を入れることができる.これを「変数に値を代入する」という.

同様に,

val2 = val1 + 35;

という文は「val2に(val1 + 35)を代入する」ことを意味する.プログラムを実行したときに, 右辺は

val1 + 35 = 10 + 35 = 45

と計算され, 「val2に45を代入する」という意味になる.

変数への代入を行った後は, それらをprintfで表示している.

printf("val1 is %d\n", val1);

前の項で説明した通り, %dはval1の値に置き換えられる.プログラムを実行したときに, val1の部分はその中身10に読み替えられ,

printf("val1 is %d\n", 10);

と同じ意味になる.

1. 変数の初期化

初期化とは, 変数の宣言と同時に値を入れることである.初期化は以下のように書く.

型 変数名 = 値;

前項のプログラムを変数の初期化で書き換えると以下のようになる.

|  |  |
| --- | --- |
| Sample1\_4.c |  |
| #include <stdio.h>  int main() {  int val1 = 10;  int val2 = val1 + 35;  printf("val1 is %d\n", val1);  printf("val2 is %d\n", val2);  return 0;  } | |

実行結果は前項と同じ.

ちなみに, もし初期化されていない変数を表示すると, どんな値が出るかわからない(この値を「不定値」と呼ぶ).何が入っているかわからないので, 初期化または代入を行っていない変数を使ってはいけない.

1. 演算

いままでのプログラムでも何気なく足し算を使っていたが, 今回はそれ以外の演算も見てみよう.

|  |  |
| --- | --- |
| Sample1\_5a.c |  |
| #include <stdio.h>  int main() {  int a = 9;  int b = 4;  printf("a + b = %d\n", a + b);  printf("a - b = %d\n", a - b);  printf("a \* b = %d\n", a \* b);  printf("a / b = %d\n", a / b);  printf("a %% b = %d\n", a % b);  return 0;  } | |

実行結果は以下のようになる.

|  |
| --- |
| a + b = 13  a – b = 5  a \* b = 36  a / b = 2  a % b = 5 |

+, -, \*, /はそれぞれ加減乗除を表す.ここでの%は, a%bという形で「aをbで割った余り」を表す.

計算結果がどんな型になるのかは, 計算に用いた値の型による.整数同士の計算なら計算は整数になる(除算のとき, 小数点以下は切り捨てられる).

[補足]

%という文字をprintfで出力したいときは,%%と書かなければならない.だから「a % b」を表示したいときは

printf("a %% b = %d\n", a % b);

と書いている.

他の計算についてみてみよう.

|  |  |
| --- | --- |
| Sample1\_5b.c |  |
| #include <stdio.h>  int main() {  int x = 0;  printf("x: %d\n", x);    x++;  printf("x: %d\n", x);  x += 5;  printf("x: %d\n", x);  x--;  printf("x: %d\n", x);  return 0;  } | |

++はインクリメント演算子という.単に値を1増やす.同様に, --をデクリメント演算子という.これは値を1減らす.x += 5は複合代入演算子の一種.その名のとおり代入と演算を一つにまとめたものである.つまり,

x = x + 5;

を省略した形.結局これは「今のxの値に5を加えよ」と述べている.x=x+5なんて,左辺と右辺が等しくないじゃないか,と思う人もいるかもしれない.しかし代入のイコールは数学におけるイコールではないことに注意しよう.右辺と左辺が等しいわけではなく, 単に「代入」するための演算子と考える.

同じような形として,

x -= a;

x \*= a;

x /= a;

x %= a;

という複合代入演算子も同様に考えられる.実はインクリメント演算子やデクリメント演算子も複合代入演算子の一種で, それぞれ,

x = x + 1;

x = x - 1;

と同じ意味である.

1. キーボードから入力

前項のプログラムを改良して, 「キーボードから入力した2つの整数に対して四則演算をした結果を表示する」プログラムを作る.

|  |  |
| --- | --- |
| Sample1\_6.c |  |
| #include <stdio.h>  int main() {  int a, b;  printf("input:");  scanf("%d %d", &a, &b);  printf("a + b = %d\n", a + b);  printf("a - b = %d\n ", a - b);  printf("a \* b = %d\n ", a \* b);  printf("a / b = %d\n ", a / b);  return 0;  } | |

実行結果は以下のようになる(*27 3*は入力例).

|  |
| --- |
| input: *27 3*  a + b = 30  a – b = 24  a \* b = 81  a / b = 9  a % b = 0 |

実はscanfについて, 現時点ですべてを理解することができない.とにかく, キーボードから入力した整数値を変数に格納するためには,

scanf("%d", &変数名);

と書くことを覚えておこう.printfと似ているが, %dは入力した数を整数値として変数に格納することを表す.カンマ後に格納先の変数を指定する.ただし, その変数の先頭に&をつけなくてはならない(なぜつけなくてはならないのかを説明するには「ポインタ」を学ぶ必要がある).

2つの整数をスペース区切りで入力するには, 次のように書く.

scanf("%d %d", &変数名, &変数名);

1. これでPractice Aは解ける？

実は解けない.まだ「文字列をキーボードから入力」することについての説明がしていないから.

文字列は「配列」と密接な関係がある.配列について説明しなくてはならない.というわけで配列について説明する.

1. 配列

変数ひとつにつき一つの値しか入れられないのだった.いくつかの変数をまとめて扱いたいときがある.それが配列の出番だ.

配列とは,例えるなら

複数の箱が一直線上に並んだもの

と理解しておけばよい.それぞれの箱には番号が書かれているので,その番号を指定して値を入れたり取り出したりする.

では早速,配列を使ったプログラムのソースを見てみよう.

|  |  |
| --- | --- |
| Sample1\_7.c |  |
| #include <stdio.h>  int main() {  int a[5];  int b[5] = {1, 2, 3, 4, 5};  a[0] = b[0] + 1;  a[1] = b[1] + 1;  a[2] = b[2] + 1;  a[3] = b[3] + 1;  a[4] = b[4] + 1;  printf("a[0]: %d\n", a[0]);  printf("a[1]: %d\n", a[1]);  printf("a[2]: %d\n", a[2]);  printf("a[3]: %d\n", a[3]);  printf("a[4]: %d\n", a[4]);  return 0;  } | |

順にみていこう.まず,

int a[5];

で「int型で,要素数5の配列aを作る」ことを宣言する.配列の宣言は次の書式であることが分かるだろう.

型 配列名[要素数];

続いて,bの配列を初期化している.配列の初期化は次の書式.

int b[5] = {1, 2, 3, 4, 5};

で,bの0番目の要素を1, 1番目の要素を2, … , 4番目の要素を5として初期化している.次の書式で初期化している.

型 配列名[要素数] = {0番目の要素, 1番目の要素, … };

ちなみに要素数より小さな数しか初期化しなかった場合,書かれていない要素は0で初期化される.つまり,

int b[5] = {1, 2, 3};

と書くと,bの3番目と4番目の値には0が入る.

「0番目の要素」という言葉に違和感を覚えた人がいるかもしれない.C言語では配列は0番目から数えるのだ.よって,要素数をNとしたら,配列の番号は0 ～ N – 1までつけられる.

続いて,プログラムではa[0]～a[4]に値を代入している.そして,その値をprintfで表示している.

今回の例では,配列の要素数が5個と少なめだったが,もしもっと要素数の大きな値,例えば10000くらいだったらどうする？同じような処理を10000個も書くのはつらい.ソースコードが長くなってしまい,きっと手が腱鞘炎になるだろう.そんな問題を解決すべく「繰り返し処理」のお話をしなければいけないのだが,それはまた別のお話.いつかまた別のときに話すことにしよう.

1. char型の配列

文字列は「文字の集まり」だった.このことを考えるなら,次のようにすれば文字列を表現できそうだ.

char str[6] = {'H', 'e', 'l', 'l', 'o', '\0'};

文字列を「char(文字)型の配列」ととらえるのである.

ここで,'\0'とはなんだろうか.これは文字列の終わりを表すための文字で,「ヌル文字」と呼ばれている(NULLとは「何もない」を意味する言葉).文字列を表現するためには,最後にヌル文字を入れることを忘れてはいけない.でもいちいち入力するのは面倒.なんならシングルクォーテーションを入力しまくるのも面倒.実はそんな問題を解決するための省略記法が用意されている.

char str[6] = "Hello";

ヌル文字も省略できたし,ダブルクォーテーション二つで済んだ.しかしまだ問題がある.今回はHelloという文字が5文字だから,ヌル文字を含めて要素数6の配列を宣言すればよいことが分かる.しかし,

askdjflaksdjflkjsknkadnlskanvl

のような文字の数を手で数えるのは辛い.また,キーボードで文字を入力するとき,何文字の入力があるかわからないときもある.実は,初期化のときは配列の要素数を省略できる.

char str[] = "askdjflaksdjflkjsknkadnlskanvl";

宣言のときは要素数を省略できないので,多めに要素数を取っておくのが良い.

char str[100];

もちろんこの配列strには100文字以上の文字を入れてはいけない.もし100文字以上入れる可能性があるのなら,もっと大きな要素数で初期化するのが良いだろう.

配列の初期化でも説明した通り,ヌル文字以降は0で埋め尽くされる.

ではこれを踏まえて,文字列を入力してそれを単に出力するだけのプログラムを書いてみよう.

|  |  |
| --- | --- |
| Sample1\_8.c |  |
| #include <stdio.h>  int main() {  char a[100];  printf("input: ");  scanf("%s", a);  printf("output: %s\n", a);  return 0;  } | |

実行結果は次のようになる.

|  |
| --- |
| input: *Hello,Everyone.*  output: Hello,Everyone. |

斜体部分は,キーボードからの入力例を示している.

要素数100の配列aを宣言した後,キーボードの入力を求めて,入力した文字列をaに入れている.

文字列を入れるとき,scanfのダブルクォーテーション内に書くのは%s.またカンマの後のaには&をつけないことに注意(理由は「ポインタ」の項で学びます).

最後にprintfで単にaの中身を表示している.文字列は%sで表示することができる.

[補足]

Hello, Worldのような入力は(scanfでは)できない.スペースで区切ると,そこで入力を取りやめてしまうので,aには”Hello,”までしか入らない.

1. これでPractice Aは解ける？

解けます.解いてみましょう.

1. 演習問題を解く際の注意点

・演算の項でも述べたが、整数同士の割り算は切り捨てになる。

・余りを求める演算%について、次の視点を持っておくことが望ましい。

・余りには周期性がある

・A % Bとは「AからBの塊をできるだけ取り除いた余り」

1. 演習

[入出力の問題]

Practice\_A

ABC068\_A

[計算問題]

ABC001\_A

ABC039\_A

ABC005\_A

ABC043\_A

ABC026\_A

ABC076\_A

ABC055\_A

ABC057\_A

ABC087\_A