1.　簡単なif文

以下の出力例になるように、解答欄のifの条件を修正せよ。

<出力例>

正解です

//<解答欄>

#include<cstdio>

int main(void){

int t = 0;

if (t == 1){

printf(“正解です\n”);

}else{

printf(“まちがっています”);

}

}

2.　マクロの使い方

　以下の出力例になるように解答欄のプログラムをマクロ(#define)を用いて修正せよ。なお、出力例の数値は必ずしも一致はしない。有効数字5桁程度で一致するようにせよ。

<出力例>

sinπ = 0.000000

cosπ = -1.000000

sin(π/2) = 1.000000

con(π/2) = 0.000000

sin(π/4) = 0.707107

con(π/4) = 0.707107

//<解答欄>

#include<cstdio>

#include<math.h>

int main(void){

printf("sinπ = %f\n", sin(PI));

printf("cosπ = %f\n\n", cos(PI));

printf("sin(π/2) = %f\n", sin(PI/2));

printf("cos(π/2) = %f\n\n", cos(PI/2));

printf("sin(π/4) = %f\n", sin(PI/4));

printf("cos(π/4) = %f\n\n", cos(PI/4));

}

3.　switchの使い方

　以下の出力例になるように解答欄のプログラムを修正せよ。

<出力例>

i = 1

iは奇数です。

i = 4

iは偶数です。

i = 5

iは素数であり、奇数です。

//<解答欄>

#include<cstdio>

int main(void){

int i ;

printf(“i = “);

scanf\_s(“%d”,&i);

switch(i){

printf(“iは奇数です。\n”);

break;

printf(“iは素数であり、偶数です。\n”);

break;

printf(“iは素数であり、奇数です。\n”);

break;

printf(“iは偶数です。\n”);

break;

printf(“選択範囲外です。1~9までにしてください。\n”);

}

}

4.　enumの使い方

以下の出力例になるように、解答欄のプログラムを修正せよ。

<出力例>

今日は日曜日です（たぶん）。

//<解答欄>

#include<cstdio>

#include<math.h>

int main(void){

enum week w = SUNDAY;

switch(w){

case SUNDAY:

printf(“今日は日曜日です（たぶん）。\n”);

break;

case MONDAY:

printf(“今日は月曜日です（たぶん）。\n”);

break;

default:

break;

}

}

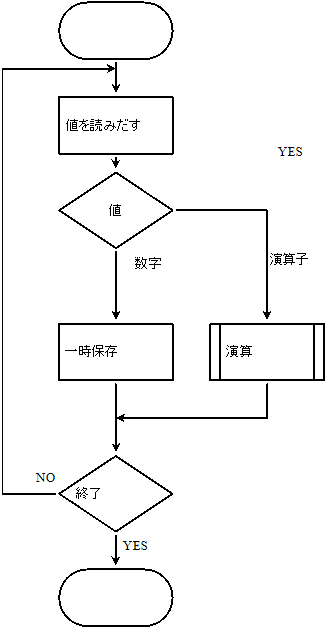
5.　複合問題　逆ポーランド記法（難しいのでやらなくともいいです）

　コンピュータにわかりやすい方法として、逆ポーランド記法というものがあります。この逆ポーランド記法の演算をソフトウェアでエミュレートします。

私たち人間がよく使う式は以下のようなものです。

　しかし、コンピューターではわかりずらいので以下のように変換されます。

　この記法を逆ポーランド記法といいます。逆ポーランド記法は以下のように演算されていきます。



以下のプログラムの穴を埋めてプログラムを完成してください。なお、100以上の値は想定しません。

//<解答欄>

#include<cstdio>

#define COMMAND\_LEN 5

enum COMMAND{

TASU = 100,

　　　　　　, //コマンドの識別子

　　　　　　,

　　　　　　,

};

int main(void){

int sp = 0; //スタックがどこまで使用しているかを示す変数

int i = 0; //コマンドを読み取る位置を示す変数

const char DATA[COMMAND\_LEN] = { 1, 2, 3, KAKERU, TASU }; //コマンドそのもの

char stack[3] = { 0, 0, 0 }; //データの一時保存領域

while (1){

switch (DATA[i]){

case TASU:

stack[sp - 2] = stack[sp - 2] + stack[sp - 1];

sp--;

break;

case HIKU:

stack[sp - 2] = stack[sp - 2] - stack[sp - 1];

sp--;

break;

case KAKERU:

stack[sp - 2] = stack[sp - 2] \* stack[sp - 1];

sp--;

break;

case WARU:

stack[sp - 2] = stack[sp - 2] / stack[sp - 1];

sp--;

break;

　　　　　: //もし数字であれば

stack[sp++] = DATA[i];

}

i++;

if (　　　　　　　　　　　　　){ //終了条件

break;

}

}

printf(“結果　%d\n”,stack[0]);

}

＊（もしできたら調べてください）

これは、CPUの動きをエミュレートしたものです。

DATA[]はプログラム領域(ROM)

stackは汎用レジスタ

iはプログラムカウンタ(PC)を表現しています。