16　文字・文字列の扱い (続き)

課題 12. 数字列を数値 (int 型) に変換する関数 myatoi() を作成し、動作例のプログラムを書いて動作を確認せよ。(符号対応・十六進版など拡張版も出来れば作ってみよ。)

解答例 12.1. 正しくない入力についてのチェックはしていない。

```
/*   myatoi.c   2010-07-05   */
#include <stdio.h>
#define MAXLEN 10
#define BASE 10

int myatoi(char *);

int main(int argc, char** argv)
{
    char str[MAXLEN];

    printf("decimal number? ");
    fgets(str, MAXLEN, stdin);

    printf("string: %s.\n",str);
    printf("value : %d.\n",myatoi(str));
}

int myatoi(char *s)
{
    int i=0, d=0;

    while( s[i] != '\0' && s[i] != '\n' )
    {
        d *= BASE;
        d += s[i] - '0';
/*
        printf("%d: %d\n",i,d);
*/
        i++;
    }

    return d;
}
```

loop の終了判定については、前回のプリントの注を参照のこと。

どうせ int 型が桁溢れするので、最大文字数は 10 に押さえた。十進法の底と区別する為に define 文を用いた。十六進版などへの拡張の際に便利?

「(一桁の) 数字→数値」の部分は関数 ctoi() などとして独立させた方が、後の拡張に便利?

loop 内でコメントアウトされている printf() を活かすと、途中経過を確認できる。

考察 16.0.2. 予め桁数が判っていなくても、この計算法で正しく計算できることを確かめよ。

この計算法は多項式の値の効率的な計算にも用いられるので、しっかり理解して頭に留めておこう。

17　main 関数の引数と返値

システムで用意されている関数や自分で作成した関数と同様、main() も一つの関数であり、やはり引数を取ったり、値を返したりする。実際、main() のプロトタイプは、int main(int, char**) と定められており、今迄もそう書いてきた。では、その引数はどこから取るのか？ そして値をどこに返すのか？

変数名は慣習で argc, argv を用いる。

C 言語では main() という名前の関数は特別で、C 言語で記述されたプログラムは関数 main() の先頭から実行を開始するという決まりになっているので、「関数 main() を呼出す」ということは、すなわち「プログラムの実行を開始する」ということに他ならない。従って、関数 main() を呼出しているのは、実行を指示するシェル (shell) ということになる。つまり、ここから引数を取り、ここに値を返すのである。

より正確には、シェルを介して OS(オペレーションシステム) に指示を出し、それに従って OS が関数 main() を呼出す、というべきか。

17–1　main 関数の引数

シェルから実行が指示されると、関数 main() は、「コマンドラインに入力された引数の数」 int argc と、「その文字列の配列へのポインタ」char **argv との 2 つを引数として取る。

argc も argv も変数なので、実行中に値を変更しても構わない。

- コマンドライン引数の数が変数 argc の初期値となる。この時、コマンドそのものも数えるので、初期値は必ず 1 以上の値を取る。
- それぞれの引数は全て文字列として char 型の配列に格納され、その各配列へのポインタ (配列の先頭アドレス) を並べて格納する配列 (要素数 argc 個) が用意され、その配列へのポインタが変数 argv の初期値となる。

argv[i] は配列名ではなく、歴としたポインタ型変数 (char * 型) なので、実行中に値を変更出来る／しても構わない。

これにより、コマンド名は argv[0]、 その次の引数は argv[1]、……、最後の引数は argv[argc-1] というように、文字列として扱うことが出来る。

各引数の各文字は argv[i][j] の形でアクセスできる。これも単なる char 型の変数なので、実行中に値を変更しても構わない。

17–2　main 関数の返値

実行が終わると、関数 main() を呼出したシェルに値 (int 型) を返す。今迄の例では省略してきたが、返値があるのであるから、main() の終了時には return 文で値を返すのが正式である。返す値は、正常終了であれば 0、異常終了の際にはそれ以外 (異常の種類によって異なる値を返すことも出来る) とするのが一般的で、それによって実行が正しく終了したかどうかを判断して、後に引き続くプログラムの実行を制御することが出来る。

実習 17.1. コマンドラインから引数を取り、引数の内容をそのまま文字列として表示。

```
/*   arg.c   2010-07-12   */
#include <stdio.h>

int main( int argc, char **argv )
{
    int i;

    printf("argc: %p (%d)\n", &argc, argc);
    printf("argv: %p (%p)\n", &argv, argv);

    for( i=0; i<argc; i++ )
    {
        printf("argv[%d]: %p (%s)\n", i, argv[i], argv[i]);
    }

    return 0;
}
```

実習 17.2. 返値の部分にマクロ EXIT_SUCCESS を用いて書き換えよ。

実習 17.3. 引数がない場合には、その旨を表示した上で異常終了 EXIT_FAILURE を返すように書き換えよ。

これも正確には、OS に値を返し、それを受け取ったシェルが解釈する、と言うべきか。

普通にプログラムの最後まで実行して終了する場合は正常終了だろうから、

```
return 0;
```

が普通だろう。この 0 も意味が解り難いので、stdlib.h には次のマクロが用意されている。

```
#define EXIT_FAILURE 1
#define EXIT_SUCCESS 0
```

プログラムの途中でいきなり return 文を書いて返ってしまって構わない。

プログラムの途中で強制的に終了 (異常終了) させて、その状態 (ステータス) を値として返すには、関数 `exit()` を用いる。例えば、プログラムを強制終了させ、そのステータスとして 1 を返したい場合は、

```
exit(1);
```

とする。他の関数から呼び出された関数内で関数 `exit()` を実行した場合にも、一挙にプログラムを終了させてシェルに値を返す。

関数の呼出側に値を返して実行を移す `return` 文とは、この点で動作が異なる。

シェルへの返値を利用する例を挙げる。これらはシェルの機能である。シェルスクリプト内で良く用いられる。

例 . 実行プログラム `./prog1` が正常終了 (返値が 0) な時に限り、`./prog2` を実行する。

```
=> ./prog1 && ./prog2
```

例 . 実行プログラム `./prog1` が異常終了 (返値が 0 以外) な時に限り、`./prog2` を実行する。

```
=> ./prog1 || ./prog2
```

課題 13 (〆切 7/18(日)). 前回の課題の `myatoi()` を用いて、コマンドラインから入力した 2 つ以上の整数の和を求めて表示するプログラム `calc` を作成せよ。

```
=> ./calc 123 45 67 8
243
=>
```

(符号対応・十六進版・不正な入力への対応など拡張版も出来れば作ってみよ。)