C言語の基礎7

コマンドライン引数

mainの引数

- これまで、mainは基本的に以下のように記述すると説明した。

```
int main(int argc, char const *argv[])
```

- しかし、以下のように書いても問題のない場合が多い

```
void main(void)
int main()
```

  - この引数の意味するものは、何だろうか？

> `argc` と `argv` は、主にCおよびC++プログラミング言語で使用される変数。
>
> `argc` は、"argument count"（引数の数）の略で、プログラムに渡されたコマンドライン引数の数を表す。コマンドライン引数は、プログラムを起動する際にプログラム名の後ろにスペースで区切られて指定される追加の情報。
>
> `argv` は、"argument vector"（引数の配列）の略で、コマンドライン引数を保持する文字列の配列。`argv[0]` にはプログラム名が格納され、`argv[1]`、`argv[2]`、...、`argv[argc-1]` にはプログラムに渡された各コマンドライン引数が格納される。

> https://www.team-aries.com/blog/osaraic-02a/

実際に試して確認してみよう

以下のプログラムを作成し、動作を確認する #1

【argc.c】

```c
#include <stdio.h>

int main(int argc, char const *argv[]){
    printf("argc : %d\n",argc);
    return 0;
}
```

【実行手順】

```
$ gcc argc.c
$ ./a.out
$ ./a.out abc 10 xyz
```

- argvはメモリ上で、どのように値が格納されるのだろうか？

  ```
  *argv[]
  ```

  は、以下のように記述する場合がある（ダブルポインタともいう）

  ```
  **argv
  ```

  - どのように考えれば良いのだろうか？
    - ポインタへのポインタを表す書き方 → 調べてみてください。

以下のプログラムを作成し、動作を確認する #2

【argv.c】

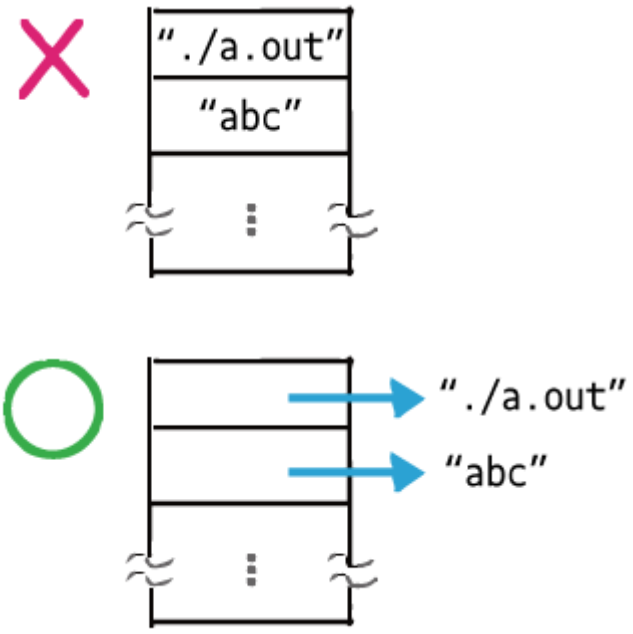
```c
#include <stdio.h>

int main(int argc, char const *argv[]) {
    printf("argc : %d\n", argc);
    for (int i = 0; i < argc; i++) {
        printf("%d : %s\n", i, argv[i]);
    }
    return 0;
}
```

【実行手続】

```
$ gcc argv.c
$ ./a.out
$ ./a.out abc 10 xyz
```

`argv[]` は「文字列を指すポインタ」を格納する配列



> https://www.team-aries.com/blog/osaraic-02a/

実行時にプログラムに対し引数を与え動作を変える

- 実行時に名前を入力し、出力するプログラム

```c
#include <stdio.h>

int main(int argc, char const *argv[]) {
    if (argc <= 1) {
        printf("名前を引数として与えてください。\n");
        printf("使い方：./a.out YourNmae\n");
    } else {
        printf("あなたの名前は %s です。\n", argv[1]);
    }
    return 0;
}
```

練習

- 以下のプログラムを作成してください。【prog1.c】
  - コマンドライン引数で、`a` もしくは `b` を与える。
  - `a` の場合は、関数 `func_a()` を実行し、`b` の場合は `func_b()` を実行する。

【prog1.c コード一部】

```
void func_a(void) {
    printf("Called func_a()\n");
}

void func_b(void) {
    printf("Called func_b()\n");
}
```

【実行例】

```
$ ./a.out
aもしくはbを与えてください
使い方：./a.out a | b
$ ./a.out a
Called func_a()
$ ./a.out b
Called func_b()
$ ./a.out c
aもしくはbを与えてください
使い方：./a.out a | b
```

補足

- プログラムを途中で終了させる場合は、`<stdlib.h>` の `exit()` 関数を使用する

> https://kaworu.jpn.org/c/C%E8%A8%80%E8%AA%9E%E3%81%AEexit%E9%96%A2%E6%95%B0%E3%81%AE%E4%BD%BF%E3%81%84%E6%96%B9

  - 正常終了→ 0 を返す
  - 異常終了→ 1 を返す（0以外）
- 「main関数でのreturn(0)とexit(0)の違いは？」
  - 上記リンクをよく読んで理解してください

mainの戻り値

- Linux OSでは、シェル（shell）というプログラム（CUIのこと）を通じて、コマンド（プログラム）の実行を行っている
    - シェルには、シェルスクリプトを用いることができる（Windowsのバッチファイルのこと）
    - 現在Ubuntuで標準的に使用されているのは `bash` というシェル。（自由に変更することが可能）
        - MacOSでは、現在は `zsh` が標準になっている
- 先の練習で作成したprog1.cをコンパイルし（a.outとする）、シェルスクリプトで実行させてみる
    - 一般的に、シェルスクリプトの拡張子は `.sh` が多いが、拡張子なしの場合もある

【compile.sh】

```
#!/bin/bash

# コンパイルする
gcc prog1.c

# 実行する
./a.out
```

【実行方法1】

```
$ bash compile.sh （もしくは sh compile.sh）
```

【実行方法2】

```
$ chmod +x compile.sh ← 実行権限を付与
$ ./compile.sh
```

mainの戻り値を確認してみよう

【check.sh】

```
#!/bin/bash

./a.out

status=$?

echo "プログラムの終了ステータス: $status"

if [ "$status" -ne "0" ]; then
    echo "==(X) プログラムが異常終了しました =="
```

```
    exit 1
fi

if [ "$status" -eq "0" ]; then
    echo "==(O) プログラムが正常終了しました ==="
    exit 0
fi
```

- `./a.out` に引数 `a` または `b` を付けて実行してみてください。

シェルスクリプトの書き方

> https://tech-blog.rakus.co.jp/entry/20210525/shellscript
>
> https://www.wakuwakubank.com/posts/347-linux-shell/

argvはすべて文字列

- 引数で渡された値はすべて文字列として格納される。
    - では、数値を与えたい場合はどうすればよいだろうか？

【実行したい例】

```
$ ./add 30 50
30+50=80
```

- そのためには、文字列→数字の変換を行う関数を用意する必要がある。

練習

- 以下のプログラムを作成してください。【prog2.c】
    - 1文字の数字文字を、数値に変換する関数 `char2int()` を作成する。
      `'5'` → `5`

【実行例】

```
$ ./a.out
1桁の数字を入力してください（1つ目）:3
1桁の数字を入力してください（2つ目）:5
3+5=8
```

【prog2.c】

```
#include <stdio.h>

int char2int(char);

int main(int argc, char const *argv[]) {
    char ch_a, ch_b;

    printf("1桁の数字を入力してください（1つ目）:");
    scanf("%c", &ch_a);
    printf("1桁の数字を入力してください（2つ目）:");
    scanf("\n%c", &ch_b); // 書式に注意

    printf("%c+%c=%d\n", ch_a, ch_b, char2int(ch_a) + char2int(ch_b));
    return 0;
}

int char2int(char ch) {




}
```

- 2回目の入力で、書式を `"\n%c"` にしたのはなぜでしょうか？

- 以下のプログラムを作成してください。【prog3.c】
    - 数字文字列を数値に変換する関数 `str2int()` を作成する
    - 正の数字のみを扱うこととする。
    - 数字以外の場合は文字 `'0'` とみなす。

```
#include <stdio.h>

int str2int(char *);

int main(int argc, char const *argv[]) {
    char str[20];

    printf("数字を入力してください:");
    scanf("%s", str);

    printf("%s => %d\n", str, str2int(str));
    return 0;
```

```
}

int str2int(char *str) {









}
```

- 上記のプログラム `str2int()` を負の数も扱えるようにしてください。【prog4.c】

【実行例】

```
$ ./a.out
12345 => 12345
+12345 => 12345
-12345 => -12345
```

```
#include <stdio.h>

int str2int(char *);

int main(int argc, char const *argv[]) {
    char str1[20] = "12345";
    char str2[20] = "+12345";
    char str3[20] = "-12345";

    printf("%s => %d\n", str1, str2int(str1));
    printf("%s => %d\n", str2, str2int(str2));
    printf("%s => %d\n", str3, str2int(str3));

    return 0;
}

int str2int(char *str) {
```

```
}
```

補足

- 文字が数字であるかどうかを調べるのに、以下のように記述した

```
if ('0' <= *str && *str <= '9') {
```

  - このような処理は、頻繁に使用するため、あらかじめ用意されている。
  - `<ctype.h>` について調べてみてください。
    どのような判定関数があるか確認してください。
    また基本的な使い方を確認してください。

    > https://ja.wikibooks.org/wiki/C%E8%A8%80%E8%AA%9E/%E6%A8%99%E6%BA%96%E3%83%A9%E3%82%A4%E3%83%96%E3%83%A9%E3%83%AA/ctype.h

- 今回作成した、 `str2int()` のような変換関数は、頻繁に使用するため、用意されている。
  - `<stdlib.h>` に関して調べ、似たような関数にどのような変換関数があるか調べてください。

コマンドライン引数版を作成する

- これまでの、練習から当初の目的である以下の処理を実装してみる（add.c）

```
$ ./add 30 50
30+50=80
```

  - 引数で渡す数値はintで収まる範囲とする。
  - 各種ライブラリの関数を理解していれば、利用しても良い。
    - 先程作成したものを利用しても作成可能。

補足

- コマンドライン引数を使用したプログラムのデバッグはどのようにすればよいだろうか？
  - `launch.json` の `args` に記述することで、コマンドライン引数をあらかじめ設定して実行することが可能
  - launch.jsonを生成できない場合は、以下の手順を実行

- メニュー「実行」→構成の追加→ `C/C++: (gdb) 起動` を選択
  - programの項目

    ```
    "program": "プログラム名を入力してください (例: ${workspaceFolder}/a.out)",
    ```

    を以下のように書き直す

    ```
    "program": "${workspaceFolder}/a.out",
    ```

  - argsの項目

    ```
    "args": [],
    ```

    を以下のように書き直す

    ```
    "args": ["30","50"],
    ```

    - これで、コマンドライン引数に `30` と `50` を設定したことにする。必ず文字列として設定
- 全部修正した結果以下のようになる。

```
{
    // IntelliSense を使用して利用可能な属性を学べます。
    // 既存の属性の説明をホバーして表示します。
    // 詳細情報は次を確認してください: https://go.microsoft.com/fwlink/?
linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "(gdb) 起動",
            "type": "cppdbg",
            "request": "launch",
            "program": "${workspaceFolder}/a.out",
            "args": ["30","50"],
            "stopAtEntry": false,
            "cwd": "${fileDirname}",
            "environment": [],
            "externalConsole": false,
            "MIMode": "gdb",
            "setupCommands": [
                {
                    "description": "gdb の再フォーマットを有効にする",
                    "text": "-enable-pretty-printing",
                    "ignoreFailures": true
                },
                {
                    "description": "逆アセンブリ フレーバーを Intel に設定",
                    "text": "-gdb-set disassembly-flavor intel",
                    "ignoreFailures": true
```

```
                    }
                ]
            }

        ]
}
```

    - ブレイクポイントを設定する
- 上記設定で、デバッグ実行したときに、ブレイクポイントが効かない場合、以下のオプションを付加してコンパイル処理を実行

```
$ gcc -g hoge.c
```

    - その後、再度デバッグを実行して確認してください。

問題

以下のプログラムを作成してください。

- コマンドライン引数で指定された文字列を逆順に並べて出力するプログラムを作成してください（prog5.c）

- コマンドライン引数で指定されたコマンド名と2つの数値を用いて計算し、結果を出力するプログラムを作成してください（prog6.c）
    - add → 2数の足し算

```
$ ./a.out add 10 20
10+20=30
```

    - sub → 2数の引き算
    - mul → 2数のかけ算
    - div → 2数の割り算
    - それ以外の文字列だった場合、エラー出力
    - 計算ができない場合も、エラー出力

- コマンドライン引数でファイル名と操作モードを受け取るプログラム（prog7.c）

- 操作モード
  read: ファイルの内容を表示
  write: コマンドライン引数で指定された文字列をファイルに書き込み
  append: コマンドライン引数で指定された文字列をファイルに追記
- 実行例

```
$ ./a.out read test.txt
# ファイルの内容を表示

$ ./a.out write test.txt "Hello, World!"
# ファイルに"Hello, World!"を書き込み

$ ./a.out append test.txt "New line"
# ファイルに"New line"を追記
```

- コマンドライン引数で文字列と操作モードを受け取るプログラム（prog8.c）
  - 操作モード
    upper: 文字列を大文字に変換
    lower: 文字列を小文字に変換
    reverse: 文字列を逆順に変換
    count: 文字列の長さを表示
  - 実行例

```
$ ./a.out upper "Hello, World!"
HELLO, WORLD!

$ ./a.out lower "Hello, World!"
hello, world!

$ ./a.out reverse "Hello, World!"
!dlroW ,olleH

$ ./a.out count "Hello, World!"
13
```