プログラミングII(前期)(3)

担当：電気情報工学科　奥村紀之

先週はバタバタしてごめんなさいm(._.)m

- 全員動くものと思っていたGit-itが動作しないとかネットワークが遅くてどうしようもないとか、想定外すぎてやられました…orz

- 今日はGitHubの登録をしたら教科書に進むので多分大丈夫です。

GitHubへの登録

- Git-itに出ているGitHubへのリンクからGitHubへ進んで下さい
  - もうアカウントを持ってるよ！という人もログインして下さい
- 指示に従ってGitHubのアカウントを作成して下さい
  - この時、特にこだわりがなければs.akashi.ac.jpのアドレスで作成して下さい
- すでにアカウントを持っていて、s.akashi.ac.jp以外のアドレスで作成している人は、プロフィールの変更でs.akashi.ac.jpのアドレスを追加しておいて下さい
- Git-it をRemote Controlに進めてください

GitHubへの登録

- GitHub Educationへ登録します
  - http://education.github.com/
- s.akashi.ac.jpのアドレスはGitHub社の好意により有料アカウントを無料で使えるようになっています
  - 何が美味しいかというと、Privateリポジトリを無制限に作成できます
- 基本的にGitHubにソースコードをアップすると公開されてしまうので授業で使っているコードとか人に見られたくないコードはPrivateで作成すると良いと思います
  - その後、公開したくなったら変更すればOKです
-

さて、少し問題です

- なぜ、Gitとかめんどくさそうなものを使って「バージョン管理」をした方が良いのでしょうか？

- Git-itを先に進めると少し分かるかもしれませんが、進める前に考えてみてください

課題1

- Git-itの残りの部分(Branches以降)を各自進めて、Verifyの結果をスクリーンショットで一つのフォルダにまとめて保存し、zipなど適当な圧縮ファイルとしてMoodleで提出して下さい
  - Gitで管理した方が良い理由もTeXでまとめてGitで管理してみるのも良いかもしれませんね(必須ではないですが、せっかくなので自分の意見をまとめてみるのも良いと思います)
- できれば、Select Directoryで「自分のフォルダだ」と分かる名前のディレクトリ下で作業してもらえると嬉しいです
  - 先週も言いましたが、「自分の手を動かす」のがプログラミング上達の基本です
  - 低学年でコピペ術を覚えてしまうと本当に伸びません
- 提出期限
  - 5/1　8:30まで(GW前ですがやっておいてください)

前回までの復習

- Gitを使ったバージョン管理方法の基礎を学びました
  - init, status, diff, commitなど基本コマンドを覚えておきましょう

- C言語の最小プログラム(？)を覚えました
  - C言語には必ずmain関数が必要
  - ブロックは中括弧で括られる
  - 文の末尾はセミコロンが必要
  - 標準入出力はstdio.hで管理
  - 整数型の返値というものが存在する

少しやってみましょう(エラー体験)

- 5分ほど時間を取るので、以下のようなことをやってみるとどうなるか試してみて下さい(コンパイルしてみて下さい)
  - #include <stdio.h> の行を消してしまう
  - int main(int argc, char **argv) を main(int argc, char **argv)にしてみる
  - どこかの行のセミコロンを消してみる
  - どこかの括弧を一つ消してみる
  - returnの行を消してみる

- 色んなエラーで怒られると思います
  - 基本的なエラーなので、「英語で表示されても読んで」下さい
  - 今後、こういったエラーを大量に見て泣きそうになりますので、どういうことをするとどんなエラーになるか、を知っておきましょう

今日のお題

- 変数

復習

- Pythonでの「変数」というのは、ものすごくフレキシブルに使えました
  - a = 100 とやれば100という数値が代入される
  - a = ‘Today is fine!’ とやれば、そういった文字列が代入される
- 配列(タプルとかリストとか)も簡単に代入できる
- 「a」ってそもそも何？ということすら気にせずに「a」が使えた

C言語での「変数」

- C言語ではPythonのようにプログラムの中でいきなり「a = 10」とか
  やっても、正しく動作しません

- テキストP.8のstart.cを改変して、以下のようなプログラムを書いてみて
  コンパイルして下さい

```
#include <stdio.h>

int main(int argc, char **argv)
{
        a = 10;
        return(0);
}
```

「型」という考え方

- C言語では、使いたい変数の「型」を事前に宣言しておかないと使えない
  - さっきのプログラムだと、整数型を示す「int」を使ってaがint型だと指定する必要有り
  - int a; という行を足してコンパイルしてみて下さい

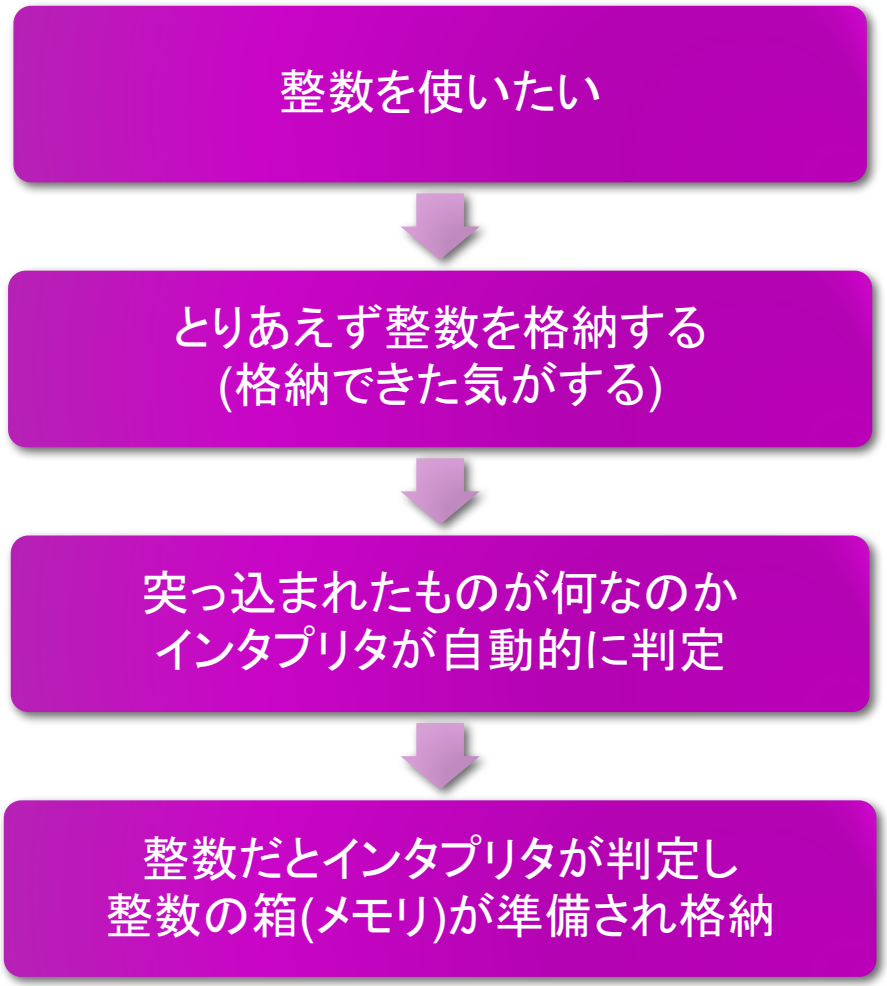
```
#include <stdio.h>

int main(int argc, char **argv)
{
        int a;
        a = 10;
        return(0);
}
```

変数を使いたいときのC言語とPythonの違い

C言語の場合

Pythonの場合



C言語における「型」

- テキストP.11の表3.1を参照
  - より詳しい「型」の説明はCクイックリファレンスの2章を参照

型	数値の酒類	サイズ
int	整数	4
char	整数	1
double	浮動小数数点(実数)	8

変数の宣言

- 先ほどサラッとやってもらっていますが、C言語では「使いたい変数を使う前までに宣言しておく」必要があります
  - さっきの例だと、a = 10; と代入する前までに int a; と宣言しておく
  - 実は変数の宣言はC99以前までは関数の先頭部分でする必要があったのですが、それじゃ使いにくいだろうってことで、使いたい変数の直前までに宣言すれば使えるように拡張されました

- 宣言の書式は　「型名 変数名;」　です
  - double d;
  - char c;
  - int* i;
  - unsigned int u;

変数名のルール

- 型と同時に宣言する必要のある「変数名」には命名規則がある(P.10)
  - 使える文字はA～Z, a～z, 0～9, アンダースコア(_)だけ
  - 大文字と小文字は区別される
  - 先頭に数字は使えない(数字だけもダメ)
  - 長さは無制限
  - 予約語は使えない(Cクイックリファレンス P.14参照)
- 使える例
  - Hoge, piyo, hoge2, _test, form
- 使えない例
  - 7up, for, x-y, 33

代入と演算

- 変数の宣言さえ忘れなければ、この辺りはほとんどPythonと同じなので、テキスト3.2章を参考にしてみて下さい

- 少し違うところと言えば、インクリメント・デクリメントがC言語では使えますので便利です
  - x = x + 1; とか x += 1; とか書くのがめんどくさいので、x++; と書けばOK
  - x = x - 1; とか x -= 1; とか書くのがめんどくさいので、x--; と書けばOK

数学的な計算

- math.hというヘッダを利用すると数学の演算(浮動小数点)ができます
  - 三角関数、指数関数、対数関数…もう習っていますかね？
  - CクイックリファレンスP.310〜P.312を見て下さい

- stdlib.hというヘッダを利用すると整数演算ができます

ここでprintfについて

- すでに「printf」という関数を使っていますが、これはPythonでいうprint文に相当します
  - Python3系で習っているならprint(‘test’)みたいな書き方をしたと思います
  - Python2系ならprint ‘test’ですね

- C言語で画面に出力したい場合は printf という関数を使います(P.6, P.14)

```
#include <stdio.h>

int main(int argc, char **argv)
{
        printf(“Hello, World. ¥n”);
        return(0);
}
```

printf の書式

- ～の書式という表現をよく出しますが、要はその関数の使い方です

```
#include <stdio.h>

int main(int argc, char **argv)
{
        printf(“Hello, World. ¥n”);
        return(0);
}
```

- printf の後は　(　)　で括ります
  - これは、C言語のどの関数でも共通です(Python3系でもそうでした)

printf の書式

- 表示したい文字列をダブルクオーテーション　”　”　で囲います
  - パワポのありがた迷惑な機能で　“”　左側が反転してますが、反転しなくて良いです

```
#include <stdio.h>

int main(int argc, char **argv)
{
        printf(“表示させたい文字列”);
        return(0);
}
```

エスケープシーケンス

- Pythonのprint文と違って、C言語のprintfでは「勝手に改行されません」
  - 明示的に改行を指示する必要がある

```
#include <stdio.h>

int main(int argc, char **argv)
{
        printf(“Hello, World. ¥n”);
        return(0);
}
```

- パワポのありがた迷惑な機能で￥マークになってますが、バックスラッシュです
- ¥nを消してコンパイル→実行してみて下さい
- また、P.14の生麦生米生卵の例を動かしてみて下さい

エスケープシーケンス

- 他にもいくつかエスケープシーケンスがあります(P.14)
  - CクイックリファレンスP.39参照

文字	意味
¥n	改行
¥t	タブ記号
¥¥	¥記号そのもの
¥”	“記号そのもの
¥'	‘記号そのもの
¥0	文字列の終端(ヌル文字)(次週出ます)

問題

- ビープ音を鳴らすためにはどうすれば良いでしょうか？

答え

- ¥a というエスケープシーケンスを使います

```
#include <stdio.h>

int main(int argc, char **argv)
{
        printf("¥a");
        return(0);
}
```

これだけだと面白くない

- Pythonでもやったように、色んな数値とかを入れた変数を表示させたい
  - まず、次のプログラムを動かしてみて下さい

```
#include <stdio.h>

int main(int argc, char **argv)
{
        int m = 4, d = 25;
        printf("今日は%d月%d日です。¥n", m, d);
        return(0);
}
```

変数への代入

- サラッと書きましたが、変数への代入は宣言してからやっても良いですし宣言時に初期値の代入もできます
  - 初期値、というのは次のプログラムを動かせば分かります

```
#include <stdio.h>

int main(int argc, char **argv)
{
        int m, d;
        printf("今日は%d月%d日です。¥n", m, d);
        return(0);
}
```

printf の書式

- 変数の値を表示させたいときは “” の間に %何とか という指定を入れて ””の右側に表示させる順(%～に対応している順)に変数名をコンマ区切りで並べていく(色ごとに対応しています)

```
#include <stdio.h>

int main(int argc, char **argv)
{
        int m = 4, d = 25;
        printf(“今日は%d月%d日です。¥n”, m, d);
        return(0);
}
```

printf の書式

- 変数をコンマ区切りで列挙する部分で演算もできます

```
#include <stdio.h>

int main(int argc, char **argv)
{
        int m = 4, d = 25;
        printf("%d×%dは%dです。¥n", m, d, m*d);
        return(0);
}
```

変換指定子

- %何とか、の「何とか」の部分には色々指定できます(P.15)
  - 色々指定できる、というのは「何を指定しても良い」という意味ではありません
  - 自分が宣言した変数の型に応じたものを指定しましょう
- 浮動小数点(double型)なら　%lf　(テキストにはないですがエルエフで)
- 文字(char型)なら　%c
- ポインタなら　%p

型と表示

- 次のプログラムを動かしてみて下さい(Warningは気にせずに)

```
#include <stdio.h>

int main(int argc, char **argv)
{
        int i, j;
        double d, e;
        i = 1;
        j = 1.5;
        printf("i = %d, j = %d¥n", i, j);
        d = 1;
        e = 1.5;
        printf("d = %lf, e = %lf¥n", d, e);
        return(0);
}
```

型と表示

- C言語は、変数の型を厳密に見ているので、int型にdouble型の値を入れても小数点以下は切り捨てられる
  - 適当に解釈して適当に型を変えてくれたりしない

- double型にint型の値を入れるとdouble型として扱われる
  - P.11の表のサイズを見ると通常はサイズの違うものに代入すると代入した先の型にサイズが整えられる
    - double(8バイト) → int(4バイト)
    - int(4バイト) → double(8バイト)

型のサイズの確認

- sizeof 演算子を使うと、型の大きさを確認できます
  - sizeof 演算子はsize_t型の演算結果を返すので、変換指定子は%zuとして下さい
  - %dでも動くのですが、厳密にやりましょう

```
#include <stdio.h>

int main(int argc, char **argv)
{
        printf("int型：%zuバイト¥n", sizeof(int));
        return(0);
}
```

課題2

- C言語で使える様々な型の大きさを表示させるプログラムを作って下さい
  - Cクイックリファレンスの2.1〜2.3章に出てくるものを試しましょう

- 円周率 pi = 3.1415 として、半径rの円周の長さと面積を求めて表示させるプログラムを作って下さい
  - このとき、テキストのP.15を読みながら、表示させる桁数を色々変えてみて何が起こるか観察してみて下さい

- 課題の提出先：Moodle
  - 作ったソースコード(何とか.c)と、その実行結果をコピペしたもの(何とか.txt)をアップロードして下さい

- 提出期限：5/8(月)　8:30まで