ソフトウェア概論 A/B

-- 関数の作り方 again --

数学科 栗野 俊一 / 渡辺 俊一 (TA: 石川 渓 / 山脇 直樹 [M1])

2019/07/12 ソフトウェア概論

伝言

私語は慎むように !!

- 出席パスワード : 20190712
- 色々なお知らせについて
  - 栗野の Web Page に注意する事

    http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino
- VNC Server Address : 10.9.154.227
  - Password : vnc-2018
- 廊下側の一列は遅刻者専用です(早く来た人は座らない)
- 講義開始前に済ませておく事
  - PC の電源を入れておく
  - ネットワークに接続しておく
  - 今日の資料に目を通しておく
- 講義前の注意
  - 講義前は、栗野は準備で忙しいので TA を捕まえてください
- やる気のある方へ

今後の予定(後ろから)

- 今後の予定
  - 2019/07/26 (講議最終日)
    - 試験 / Note-PC 必須 / PC のトラブル対応はしない / 課題提出最終日
  - 2019/07/19 (講議最終日前)
    - 前期のまとめ(落穂拾い) / 模擬試験 / Note-PC 必須 / 環境を整える
  - 2019/07/12 (本日)
    - 関数の作り方 again

前回(2019/07/05)の復習

- 前回(2019/07/05)の内容 (二週目に入った)
  - 「Hello, World」again (お呪いが減った)
  - 順接

お知らせ

- 本日(2019/07/12)の予定
  - 制御構造
  - 関数(作成方法/引数)
  - データ型
- 本日(2019/07/12)の目標
  - 講議
    - 関数と三つの制御構造(順接/分岐/繰返[再帰])
    - 関数の値
  - 演習
    - 課題の提出

今週 (2019/07/12) の課題

- 今週 (2019/07/12) の課題
  - 課題 20190712-01:
    - ファイル名 : 20190712-01-QQQQ.c (QQQQ は学生番号)
    - 内容 : 二つの整数の積を返す関数
    - ファイル形式 : テキストファイル(C 言語プログラムファイル)
  - 課題 20190712-02:
    - ファイル名 : 20190712-02-QQQQ.c (QQQQ は学生番号)
    - 内容 : 自然数の階乗を返す関数
    - ファイル形式 : テキストファイル(C 言語プログラムファイル)
  - 課題 20190712-03:
    - ファイル名 : 20190712-03-QQQQ.c (QQQQ は学生番号)
    - 内容 : 二つの非負の整数の最大公約数を返す(ユークリッドの互除法)
    - ファイル形式 : テキストファイル(C 言語プログラムファイル)

先週 (2019/07/05) の課題

- 先週 (2019/07/05) の課題
  - 課題 20190705-01
    - ファイル名 : 20190705-01-QQQQ.c (QQQQ は学生番号)
    - 内容 :「Hello, 自分の名前」を出力する
    - ファイル形式 : テキストファイル(C 言語プログラムファイル)
  - 課題 20190705-02:
    - ファイル名 : 20190705-02-QQQQ.c (QQQQ は学生番号)
    - 内容 :「Hello, 自分の名前」を三回出力する
    - ファイル形式 : テキストファイル(C 言語プログラムファイル)
  - 課題 20190705-03:
    - ファイル名 : 20190705-03-QQQQ.c (QQQQ は学生番号)
    - 内容 : コマンドライン引数に名前を一つ指定し、その人に挨拶をする
    - ファイル形式 : テキストファイル(C 言語プログラムファイル)

「関数」という考え方 (復習)

- 関数の定義とは(What)？
  - 「プログラムの一部」に「名前」を付ける事
    - 「名前」を「関数名」と呼ぶ
    - 「プログラムの断片」を「関数の本体」と呼ぶ
- 関数をどうやって利用する(How to use)？
  - 「関数名」を指定するだけで「関数本体」が実行される(関数呼出し)
- 関数を定義する理由は (Why)？
  - 「プログラムの断片」に「『名前』が付けられる」ので、分かり易い
    - もちろん、「断片の内容に対応した『分り易い名前』を付ければ..」だが..
  - 「関数名前」を指定するだけで「関数本体」が実行される
    - 何度も同じ事をする場合に便利(プログラムが短くなる)
  - 「引数」を利用する事により「色々な断片」を「一つの関数本体」にまとめられる
    - 何度も似たような事をする場合に便利(プログラムが短くなる)
  - 一箇所の「関数本体」を直すだけで、多数の場所の命令を直す効果がある
    - 「『コピペ』がバグの増殖を促す」事を防ぐ事ができる

「関数」の表現方法 (復習)

- □ 関数定義(の文法)
  - ○「関数定義」は、「関数頭部」と「関数本体」に分けられる
  - ○「関数頭部」は、「関数宣言」「関数名」「仮引数宣言」に分けられる
    - ▹「関数宣言」は、void(これまで)/int(main だけ)/型名(値を返す場合)
    - ▹「関数名」は、自由に決めて良い(他と重複すると駄目だが..)
    - ▹「仮引数宣言」は、「(」+「仮引数宣言並び」+「)」
    - ▹「仮引数宣言並び」は、「void」(無い場合)か、「型名 変数名」のカンマ(,)区切
    - ▹「関数本体」は、「{」+「命令列」+「}」
- □ 関数呼出し(の文法)
  - ○「関数呼出し」は、「関数名」+「実引数並び」
  - ○「実引数並び」は、'()' か、'(' +「式」のカンマ並び + ')'

色々な型のデータを引数に持つ関数と型宣言 (復習)

- 最初の引数付き関数
  - 引数がないか、文字列を引数としていた
    - 「char *」をお呪いとし、関数を呼び出す時に、文字列を指定
    - 変数には文字列が入っているとして、考える
- 色々な型のデータ(文字,整数)を引数に持つ関数の場合
  - 引数宣言にデータ型(文字の場合は「char」、整数の場合は「int」)を指定する必要がある
- 型宣言
  - 「char *」/「char」/「int」は、実は、「引数の型」を表現していた
    - 「char *」は「文字列」/「char」は「文字」/「int」は「整数」
  - 変数に、その型と異る値を入れようとすると「エラー」になる
- 「型」と「演算」
  - 「文字列」に「1 を加える」と、「短くなった文字列」
  - 「文字」に「1 を加える」と、「次の文字」
  - 「整数」に「1 を加える」と、「一つ増えた整数」
    - 同じ「1 を加える」という「演算」でも、「意味」が異る
  - 「演算」と「型」は「一組」で考える必要がある

データ(Data)とコード(Code)

- □ データ(Data) : 値を持つもの
  - ○ 即値(定数値/定数表現), 変数, 関数呼出し, 式
    - ▹ 関数の「実引数」として「渡せる」物
    - ▹ 型を持つ / 計算できる / 入出力できる / 関数の値として返せる
- □ コード(Code) : 値を操作するもの
  - ○ (今の所は..) 関数呼出し (しか、学んでいない)
    - ▹ 関数の本体部分に記述するもの
    - ▹ 関数の「実引数」としては、渡せない (記述しても渡されるのは関数の返値)

構文 : (単純な)コードから(複雑な)コードを作る仕組

- ○ 順接 (+ ブロック) / if 構文
- □ 関数定義 : コードに名前を付ける仕組
  - ○ 「関数を定義する」事により、その関数の「関数呼出し」が可能となる
    - ▹ 関数は予め定義して(コンパイルして)置かないと、利用できない
    - ▹ printf 等の libray (ライブラリ)関数は、既にコンパイル済の物を利用している
    - ▹ リンク(Link)によって、複数の関数が「纏め」られて、実行ファイルが作られる

関数の返り値と return 命令 (再)

- □ void 型関数
  - ○ 関数の前に void が付けられている
    - ▹ 実は、C 言語でも、「特別(void 型)」な「関数」
    - ▹ cf.「数学の関数」とは違う
  - ○「数学の関数 f(x)」: x として f に何かを与えると f(x) という値が得られる
    - ▹ cf. f(x)=x^2 なら f(3)=3^2=9, sin(π/6)=1/2
- □ 非 void 型関数
  - ○ 関数の前に「関数値の型」が付ける
    - ▹「数学の関数」と同じように「値(返り値)」を計算して「返す」事ができる
    - ▹ cf. int f(int x) : 整数型の値を与えると整数型の値を返す関数
- □ 値を返す関数の作成方法
  - ○ 関数の前の型宣言は、関数の返す値の型を書く
  - ○ 関数が返す値は、return 命令の後ろに書く
    - ▹ return 命令が実行されると、その時点で関数が終了する事に注意

様々な文字

- 一般的な文字の表現
  - 文字自身を表す記号を「'」(シングルクォート)で挟む
    - 例：「A」を表す「文字」は、「'A'」で表す
- 様々な文字：対応する文字記号が存在しない場合
  - '\n'：改行を意味した、'\0'：End of String を意味した
    - 「対応する文字記号」が(キーボード上に..)存在しない場合は、「特別な表記」が必要
- エスケープシーケンス
  - 「エスケープ」: 脱出 -> 通常からの「脱出」 -> 特別な意味を持つ
    - 「文字」を表現する場合に「\」がある場合は「エスケープ」される
    - 'n' (単なる「n」) と '\n' (「改行」を表す) は異る
    - 先行する「\」(エスケープ記号)によって、「異る意味付け」がなされる
- エスケープ表現の例
  - '\n' -> 改行 / '\0' -> EOS / '\\' ->「\」自身 / '\"' ->「"」 / '\'' ->「'」