プログラミング基礎 #07
値渡しと参照・ポインタ

担当: 向井 智彦

先週のおさらい

戻り値の型名　関数名（引数リスト）｛　処理;　｝

- 関数
  - 処理のひとまとまり
  - 数値を受け取って（引数）、出力する（戻り値）
- 関数の戻り値と引数
  - 戻り値は1つ以下、引数は0個以上
  - なにも無い場合は void
  - 配列は引数に指定できる、戻り値にはできない
- 関数プロトタイプ宣言とヘッダファイル

値渡しとは？

- 変数が持つデータの内容を，別の変数にコピーして渡す操作
  - コピー先で値が書き換えられても，コピー元には影響を及ばさない
- 代入演算「=」の挙動
- 関数の引数の挙動
- 関数の戻り値の挙動

値渡しの動作確認 (1/2)

```
int main()
{
   int x = 10;
   int y = x;
   x = 5;
   cout << x << “, ” << y << endl;//?
}
```

値渡しの動作確認 (2/2)

```
void ZeroClear(int a) {
   a = 0;
}
int main() {
   int x = 10;
   ZeroClear(x);
   cout << x << endl; // ?
}
```

参照とは？

- 同一データ/同一変数に別名を与える処理

```
int main()
{
  int x = 10;
  int &y = x;  // int型変数への参照
  x = 5;
  cout << x << ", " << y << endl;
  y = 8;
  cout << x << ", " << y << endl;
}
```

参照渡しとは？

- 参照を通じた関数への引数渡し

```
void ZeroClear(int &a) {
  a = 0;
}
int main() {
  int x = 10;
  ZeroClear(x);
  cout << x << endl; // ?
}
```

戻り値の代替としての参照渡し

- 関数の出力を受け取るための参照引数

```
void Double(int &output1, int input) {
  output = input * 2;
}
int main() {
  int x = 10;
  int y = 0;
  Double(y, x);
  cout << x << ", " << y << endl;
}
```

参照の特徴

- 変数のように後から上書きできない

```
int main() {
  int x = 10;
  int &y = x;  //yはxの別名
  int z = 0;
  x = 5;
  cout << x << ", " << y << endl;
  y = z;  //値の代入(≠参照先の変更)
  cout << x << ", " << y << endl;
}
```

参照を使うケース

- 引数として渡したデータの内容を関数側で上書きするとき
  - 複数の戻り値→複数の参照渡し
- 巨大なクラスを関数の引数とするとき
  - 値渡しするとコピー/クローンの計算時間が増大
  - 参照渡しだと「別名」を作る処理のみ

ポインタ

- 別名の付け方 その2

```
int main()
{
  int x = 10;
  int *y = &x;  //int型変数へのポインタ
  x = 5;
  cout << x << ", " << *y << endl;
  *y = 8;
  cout << x << ", " << *y << endl;
}
```

ポインタを通じた参照渡し

- 参照を通じた関数への引数渡し

```
void ZeroClear(int *a) {
  *a = 0;
}
int main() {
  int x = 10;
  ZeroClear(&x);
  cout << x << endl; //?
}
```

戻り値の代替としてのポインタ参照渡し

- 関数の出力を受け取るための参照引数

```
void Double(int *output, int input) {
  *output = input * 2;
}
int main() {
  int x = 10;
  int y = 0;
  Double(&y, x);
  cout << x << ", " << y << endl;
}
```

ポインタ変数の特徴

- 通常の変数のように後から上書きできる

```
int main() {
  int x = 10;
  int *y = &x;  //yはxの別名
  int z = 0;
  *y = 5;
  cout << *y << ", " << z << endl;
  y = &z;  //参照先の変更
  *y = 7;
  cout << *y << ", " << z << endl;
}
```

ポインタを使うケース

- ライブラリ/APIがポインタ使用を想定する場合
- 実行中に参照先を変更する必要がある場合
- プログラム実行中にクラスインスタンスを作る必要があるとき
  - プログラミング中に何個の変数を用意すべきか（≒配列の長さが）わからない場合
  - new/new[] & delete/delete[] （講義外）

ポインタの正体

- オブジェクトのメモリ位置（アドレス）
  - int *p = &a; は，int型変数 a のメモリ位置
  - 配列 int a[]; の a は，先頭要素へのポインタ
- 参照渡し&ポインタ渡し
  - オブジェクトのメモリ位置情報を渡し，その位置が指す変数を直接読み書き
  - 一方，値渡しは変数の内容をコピー/クローン

ポインタの正体

```
int main()
{
  int value = 0;
  int *ptr = &value;
  int array[5] = {0, 1, 2, 3, 4};
  int *arrayptr = array;
  char str1[7] = “hello”;
  char *str2 = str1;
}
```

特殊なポインタ : nullptr

- ヌルポインタ
- どの変数/配列も指していない

```
int *a = nullptr;
int b = 0;
*a = 0; // エラー
a = &b;
if (a != nullptr) { //nullチェック
  std::cout << *a;
}
```

まとめ

- 値渡しと参照渡し
  - 値渡し: 変数が持つ値を別の変数に代入
    - 代入先で変更が生じてもコピー元はそのまま
  - 参照渡し: 変数に別名を与える
    - 参照代入先で変更が生じるとコピー元にも影響
- ポインタ，配列とポインタ
  - ポインタ: 変数に別名を与える方法2
  - ポインタが指す変数は変更可能（参照は変更不可）
  - 配列名＝ポインタ
- 特殊なポインタ nullptr: 何も指さない

演習課題07-1： sort2 (ポインタ)

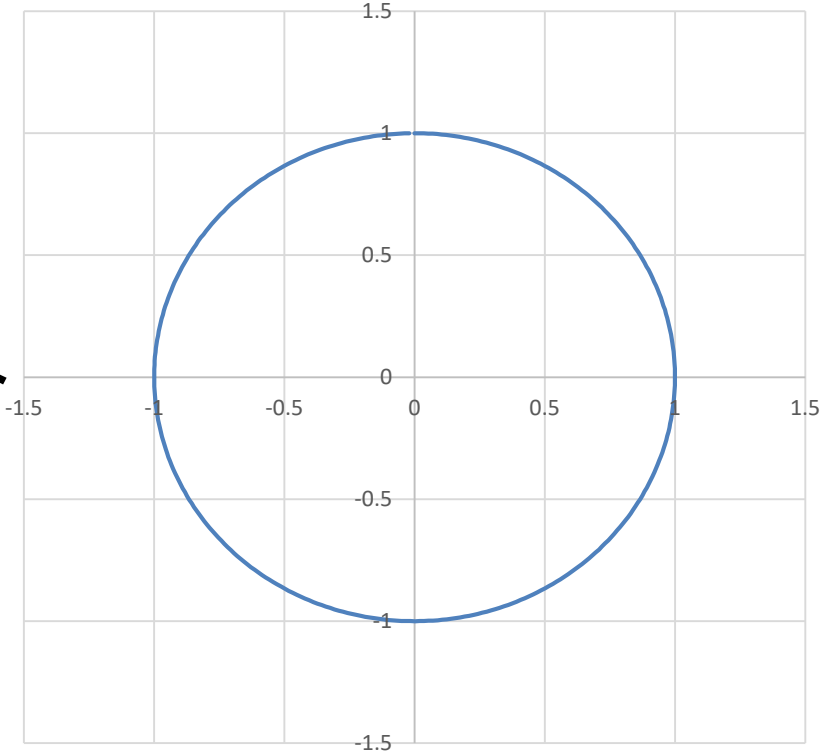
提出期限：11/19（月）、ファイル名： 07-01.cpp

- 2つの整数値を小さい順に並び替える関数を完成させる
  - ポインタを利用して並び替え結果を戻す

演習課題07-2：sincos (参照)

提出期限：11/19（月）、ファイル名： 07-02.cpp

- 指定された角度 (degree) の正弦と余弦を同時に計算して戻す関数 sincos を実装し、0°から360°まで1度刻みに計算結果を出力
  - 参照を利用して2つの計算結果を戻す
  - 出力結果を「.csv」ファイルにコピペし、Excelでプロットすると円が描ける(はず)



演習課題07-2ex：リサージュ

提出期限：11/19（月）、ファイル名： 07-02ex.cpp

- 07-02で作成した関数 sincos を拡張し、リサージュ図形（Lissajous）を出力するプログラムを作成
- wikipedia: "リサジュー図形"
  - 角度に加えて a、b、δ を Lissajous関数の引数として追加
  - 好みの図形が出力される状態に設定して提出