ソフトゼミC
第二回
C++の基礎

2013/08/06

エレクトロニクス研究部

今日は何をするの？

- C++とは何か？
- ストリームライブラリを使った
  入出力cin/coutについて
- CとC++の構造体の違い
- classの基礎とメンバ関数、カプセル化
- コンストラクタとは

C++とは？

- C++とは何か？

C++はその名の通りC言語の
拡張として1983年に作られた。
開発当時は「C with Classes」と呼ばれ、C言語にクラスの概念を持たせた言語である。
C言語の文法とSimulaのオブジェクト指向の概念を
兼ね備えた言語である。
ちなみにMicrosoftはC++を魔改造し、C++++ならぬ
C#という言語を生み出したりしている。

CとC++の違い(1)

- C++で拡張された機能
- 変数の宣言が先頭以外でも出来るように

```
int main(void){
  int a;
  a=20;
  int b[10];  //文の途中でも変数が宣言できる。
  for(int i=0;i<10;i++){
  //for文の中でも変数の宣言が出来る。
    b[i]=i;
  }
  return 0;
}
```

CとC++の違い(2)

- 型の省略が可能に

```
struct TE{
  int a;
  int b;
};
int main(void){
  TE te;     //structを省略できる。
  te.a=10;
  te.b=20;
  return 0;
}
```

構造体(struct),共用体(union),列挙型(enum)に対して
typedefを使わなくても型を省略できるようになる。
※共用体、列挙型についての説明は時間があればやります。

CとC++の違い(3)

- boolean型の追加

boolean型とは真(true)か偽(false)の2つの値しか
持たない1bitの変数
C++ではbooleanではなくbool型なので注意
bool b1 = true; bool b2 = false; のように使用

- **C++標準ライブラリ**

C言語にも<stdio.h> <stdlib.h> <math.h>のような標準
ライブラリが数多くあるが、C++にもC++向けの標準ライブラリが存在する。中でもStandard Template Library (STL) と
呼ばれるライブラリは文字列の入出力やリストの設計を容易にすることが出来る。
詳しくはググるかC++の本を読んでみるといいよ

CとC++の違い(4)

- クラスの概念

これが一番大きい。C言語の構造体を更に拡張したもので、構造体の中の変数を外部から直接いじれないように保護したり、構造体の中でさらに関数が使えるようになったと考えるとわかりやすい？

クラスの外でメンバ(クラスの中で定義された変数)を変更できなくすることをカプセル化と言い、オブジェクト指向を構成する概念の一つである。

ストリーム入出力(1)

- ストリーム出力 cout

C++で追加されたSTLの内、代表的な入出力ライブラリを使ってみる。この時<stdio.h>ではなく<iostream>ヘッダを使うので注意！

```
#include <iostream>
int main(){
  std::cout << “Hello World!¥n”;
  return 0;
}
```

→Hello World!

ここで、”Hello World!”という文字列をstd::coutに流し込んでいる。(流し込むモノ) >> (流し込む先) と覚えると覚えやすい。

ストリーム入出力(2)

- 名前空間

std::coutというのはstdという名前空間にある
coutという関数を使っている。
stdと指定しているのはどこのcoutか分からなくなるため
しかしながら毎度毎度std::coutと書くのは面倒くさい
→using namespace std;　と先頭に記述する
この宣言以降は名前空間stdにアクセスする時
スコープ演算子(::)を省略する。という意味
※ただし、名前空間の意義を破壊する書き方なので
あまり多用するのはよくない

ストリーム入出力(3)

- 説明だけではアレなので例

```
#include<iostream>
using namespace std;
int main(){
        int a=10;
        cout << "a:" << a << endl;
        return 0;
}
```

→a:10

endlで改行して出力をするという意味

このように入出力ストリームで書式や処理を行う関数を

マニピュレータと言う

ストリーム入出力(4)

- マニピュレータ

C++の出入力はマニピュレータを使って処理をすることができる。

マニピュレータ	概要
endl	改行して出力
flush	改行せず出力
oct	8進数出力
dec	10進数出力
hex	16進数出力
setw(n)	n文字出力 %ndに相当
setfill(n)	空白部分に埋める文字を指定

setw()とsetfill()は<iomanip>をインクルードする必要がある。

ストリーム入出力(5)

- ストリーム入力 cin

入力の場合はC言語でいうとscanfに相当する。

```
int a;
cin >> a ;
```

これだけでaに値を読み込むことが出来る。

文字列の入力が簡単にできるようになる。

```
char name[20];
cout << “名前を入れてください” <<endl;
cin >> name;
cout << “あなたの名前は” << name << endl;
```

練習問題1

- 出入力ストリームを使って、値を2つ読み込み、合計値を10進数と16進数の2つで出力せよ。
- 答え

```
#include<iostream>
using namespace std;
int main(void){
        int a,b;
        cin >> a >> b;
        cout << "10進数:" << a+b <<endl;
        cout << "16進数:" << hex << a+b << endl;
        return 0;
}
```

練習問題2

- 出入力ストリームを使って、九九の表を出力するプログラムを作れ、ただしsetwを使って書式を合わせること
- 答え

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(void){
        for(int i=1;i<10;i++){
                for(int j=1;j<10;j++){
                        cout << setw(3) << i*j << flush;
                }
                cout << endl;
        }
        return 0;
}
```

C++の構造体(1)

- C++では構造体の中で関数が使える。(10.cpp)

```
#include<iostream>
#include<string.h>
using namespace std;
struct Person{
        char name[20];
        int age;
        void show();
};
void Person::show(){
        cout << "Name:" << name << endl << "Age:" << age << endl;
}
int main(){
        Person a;
        strcpy(a.name,"Tanaka");
        a.age = 19;
        a.show();
        return 0;
}
```

C++の構造体(2)

- さっきのコードの解説

構造体の中で関数を宣言し、実態は構造体の外で
定義している。Person::show()とはPerson構造体の
関数show()という意味。show()だけだと何処のshow()
か分からなくなってしまう。このような構造体の中の関数を
メンバ関数と呼ぶ

strcpy(a,b);というのはbを先頭ポインタにする文字列をaを先頭ポインタにする文字列にコピーするという意味
宣言の時以外ではa.name = “Tanaka”;という書き方は出来ないので、関数を使って代入していると思って
くれればいい。これはC言語でも使える。

クラスの初歩(1)

- C++においてクラスと構造体というのはほとんど同じ。違いは構造体がデフォルトでpublicなのに対しクラスはデフォルトでprivateである。

- public(パブリック)というのはクラスの外からでも参照できること。private(プライベート)は逆にクラスの外からでは参照できない。
- 他にprotected(プロテクティッド)というのもあるがこれは明日説明する。

クラスの初歩(2)

- なぜクラスの外から参照できなくするか・・・

長いコードを書く時に間違えて値を書き換えてしまうことがある。
このようなミスを無くすためにはそもそも外部から書き換えなくさせてしまえばいいからである。
このようにクラスの内部情報を外部から読み取れなくすることをカプセル化と言う。

クラスの初歩(3)

- 簡単なクラス(11.cpp)

```
#include<iostream>
using namespace std;
class Test{
private:
        int x;
        int y;
public:
        void set(int a,int b){
                x=a;
                y=b;
        }
```

クラスの初歩(4)

- 簡単なクラス(続き)

```
        void show(){
                cout << "x:" << x << endl;
                cout << "y:" << y << endl;
        }
};
int main(){
        Test a;
        a.set(10,20);
        //cout << a.x << endl; //エラーになる
        a.show();
        return 0;
}
```

クラスの初歩(5)

- 解説

privateで保護されているx,yには直接アクセスはできない。

値を入れるときには公開しているset関数を

表示するときにはshow関数を使う。

試しにコメントアウトを解除し直接アクセスしてみようとするとエラーになる。

練習問題3

- 時・分を要素に持つクラスTimesを設計せよ。

メンバ関数として時間をセットする関数set

時間を10分進ませる関数adv,時刻を表示する関数showを実装せよ。

また、要素である時・分はクラスの外部から書き換えられないように保護すること。

→答えは長いので別紙に(12.cpp)

コンストラクタ(1)

- コンストラクタとは？

→クラスを作り出した時に自動的に呼び出される特殊な関数。戻り値が存在しない
初期値を設定する時に使う。試しに練習問題3にコンストラクタを付けてみる。

```
class Times{
        int hour;
        int minute;
public:
        Times(int,int);  //これがコンストラクタ
        void adv();
        void show();
};
```

コンストラクタ(2)

- Set関数をコンストラクタに変更

```
Times::Times(int h,int m){ //コンストラクタの実態
        hour=h;
        minute=m;
}
void Times::adv(){
…(略)…
}
void Times::show(){
        cout << hour << "時" << minute << "分" <<endl;
}
int main(){
        Times a(23,40);  //Timesクラスの宣言時にコンストラクタが呼ばれる。
…(後略)…
```

コンストラクタ(3)

- コンストラクタの名前

コンストラクタの名前はクラスの名前と同一である。
Timesクラスであればコンストラクタは Times()となる。

- 引数

コンストラクタはこのように引数を受け取ることも出来る。
もちろん引数がなくても構わない。
引数がない場合は通常時と同じように宣言する。

デストラクタ(1)

- デストラクタとは

コンストラクタがクラスをインスタンス化(実体化)
した時に呼び出されるのに対しデストラクタは
解体時に読み出される。
デストラクタの名前はクラス名の先頭に
”~”(チルダ)を付けたものになる。
Timesクラスのデストラクタは~Times()となる。

デストラクタ(2)

- デストラクタの例

```
#include<iostream>
using namespace std;
class Test{
public:
        Test(){
                cout << "コンストラクタが読み出されました。" <<endl;
        };
        ~Test(){
                cout << "デストラクタが読み出されました。" <<endl;
        };
};
void func(){
        Test a;
}
```

デストラクタ(3)

- デストラクタの例(続き)

```
int main(){
        cout << "func関数を読み出します。" << endl;
        func();
        cout << "func関数を読み出しました。" << endl;
        return 0;
}
```

- 出力結果

func関数を読み出します。
コンストラクタが読み出されました。
デストラクタが読み出されました。
func関数を読み出しました。

デストラクタ(4)

- 解説

コンストラクタが読み出されるのはクラスをインスタンス化した時、つまりTest a;でクラスを定義した時に呼ばれる。
一方でデストラクタはインスタンスのスコープが切れる時
つまり変数の有効範囲外に出た時に変数が
使えなくなるのと同じで、クラスの有効範囲外に出て
読みだしたクラスが使えなくなる時に読み出される。

練習問題4

- 3人の名前、年齢をコンストラクタで読み込み、名前と年齢を表示させる関数show()を持つクラスPersonを設計せよ。
- ただし、名前と年齢はクラスの外部から直接参照できないようにすること。

→答えは長いので別紙に(15.cpp)

参考にしたサイト

- WisdomSoft（旧）

http://wisdom.sakura.ne.jp/

C/C++を基礎から学びたい時にオススメ

C#/Java/PHPだけでなくD言語やPerlも学べる。

- Programing Place

http://www.geocities.jp/ky_webid/

C/C++に特化している。STLについて学びたい時にはオススメ

- ためになるホームページ

http://www.booran.com/

上2つと比べるとやや取っ付きにくい

やや中級者向けで、それなりに詳しい仕様がわかる

THE END

明日はクラスについて更に踏み込んだ内容を学びます。
・コンストラクタのオーバーロード
・クラスの継承
・継承したクラスでのオーバーライド
・多重継承
をやる予定です。
場所は変わって5203になるので注意してください