知能機械プログラミング

担当者：張 斌　　Email: zhangbin@kanagawa-u.ac.jp

到達目標

本科目では、機械を知能化する技術として、プログラミング言語基礎、知能機械基礎を学修すると同時に、実習を通じて、機械を知能化する情報処理手法をプログラミングで実現するスキルの修得を目指す。将来は、人工知能技術を生かし、知能機械の実現を目指す。

授業内容

プログラミング言語の基礎を習得し、続いて基礎な知能機械に関する知識を学ぶ。機械を知能化する情報処理手法をプログラミングで実現する。

授業方針

授業はプログラム作成の基礎知識を学ぶ「講義」と、それを利用して実際にプログラムを作成する課題に取り組む「演習」で構成する。また、毎回の講義の最初の 10 分は前回の復習として、プログラミングの課題を完成してもらう。

授業実施形態

全回面接

授業運営

授業内容を必ずメモを取る。

また、毎回の講義の最初の 10 分間で、前回分について復習をし、課題を完成してもらう。

評価方法

中間テスト３０%、最終テスト７０%のトータル１００点満点で評価する。

出席状況は評価の対象としないが，不可抗力の場合でも欠席数上限は 3 回である。

オフィスアワー

水曜日 13：00～16：00 まで 12 号館 24 号室。また、講義終了後、講義室で受け付けると共に、メールで随時に受け付ける。メールアドレスは初回授業の時に示す。

授業計画

1. ガイダンスとプログラミングの試作

   シラバスに沿って全体像を把握する。データの入出力と数学演算を含むプログラミングを試作する。
2. 処理の流れを分ける if-else 文の利用法

   講義と演習により、if-else 文の利用法を学ぶ。
3. 処理を繰り返す for 文の利用法

   講義と演習により、for 文の利用法を学ぶ。
4. 処理を繰り返す while 文の利用法

   講義と演習により、while 文の利用法を学ぶ。
5. アルゴリズム設計

   アルゴリズム設計方法を学ぶ。

6. アルゴリズム設計及びプログラミング演習
第 5 回の内容を続いてアルゴリズムの設計方法を学び、演習プログラミングの作成を行う。
7. 中間テスト及び解説、質疑対応
第 6 回までの内容の理解度を確認する。
8. ポインター基礎
講義と演習により、ポインターの利用法を学ぶ。
9. クラス基礎
講義と演習により、クラスの利用法を学ぶ。
10. 知能機械基礎〜ロボットビジョン①〜
ロボットビジョンに関する基礎知識を学ぶ。特に、画像処理に関する手法を学ぶ。
11. 知能機械基礎〜ロボットビジョン②〜
ロボットビジョンの課題（画像処理）をプログラミングで実現する。
12. 知能機械基礎〜ロボットビジョン③〜
ロボットビジョンに関する基礎知識を続けて学ぶ。特に、動画処理に関する手法を学ぶ。
13. 知能機械基礎〜ロボットビジョン④〜
ロボットビジョンの課題（動画処理）をプログラミングで実現する。
14. 期末テスト及び解説、質疑対応
本講義全ての内容の理解度を確認する。

1. プログラミング言語・特徴

C

C++

Python

Matlab

Java

…

1.1 C++と C の関係・違い

1.2 関数を書く

データの型　名前（）

{

　　内容；//様々な命令は；で分ける

}

例：

```
#include<iostream>
using namespace std ;
void main ()
{
　　cout<< “hello!”; // “文字列（言語と関係ない、スペースを含む）をそのまま出力”
　　cout<< endl; //次の行に移動
}
```

1.3 データの型

int	整数	32bit
double	小数	64bit
float	小数	32bit
char	アルファベット（文字）	8bit

1.4 変数

```
#include<iostream>
using namespace std ;
void main ()
{
  int a;
  a=1; //1.4 の場合は？
  cout<<a;
}
```

1.5 四則演算、剰余(割り算のあまり)、ロジック演算

＋　-　*　/

%

and or not　&&　||　！

1.6 入力と出力　cin cout

連続で入力・出力の場合

演習問題：

(1) 下記のように出力せよ。

```
*
**
***
 **
  *
```

(2)3+5*4-8%2 をプログラミングで計算し出力せよ。

(3)a+b-c をプログラミングで計算し出力せよ。ただし、a=1.4,b=2.5,c=3 をキーボードから入力する。

2. if 関数

2.1

```
if(条件)
 {
   内容;
}
```

2.2

```
 if(条件)
{
   内容 1;
}
else   //省略可
{
   内容 2;
}
```

2.3

```
 if(条件 1)
{
   内容 1;
}
else if(条件 2)//任意個
{
   内容 2;
}
else   //省略可
{
   内容 3;
}
```

演習問題：

(1) 任意の成績を入力し、それに合わせた評価レベル（秀、優、良、可、不可)を出力せよ

3. ループ関数：for 関数

for(変数の定義；条件；追加実行)
{
　内容；//繰り返し実行
}
注：省略可の内容

演習問題

(1)1+2+3+…+10 をプログラミングで計算し出力せよ。
(2) 1*2*3*…*10 をプログラミングで計算し出力せよ。
(3)100 以内の偶数を出力せよ。
(4)100 以内の奇数の和を計算し出力せよ。
(5)100 以内の 3 と 5 の公倍数を出力せよ。
(6) 100 以内の 3 または 7 倍数を出力せよ。

4. ループ関数：while 関数

```
while(条件)
{
   内容；//繰り返し実行
}
```

注：　break と continue 命令

演習問題(while 関数を用いて実現せよ)

(1)1+2+3+…+10 をプログラミングで計算し出力せよ。

(2) 1*2*3*…*10 をプログラミングで計算し出力せよ。

(3)100 以内の偶数を出力せよ。

(4)100 以内の奇数の和を計算し出力せよ。

(5)100 以内の 3 と 5 の公倍数を出力せよ。

(6) 100 以内の 3 または 7 倍数を出力せよ。

5. アルゴリズム設計

配列　int a[5]

演習問題

(1) 配列｛4,1,5,3,2｝を大きい順で出力せよ。

6. アルゴリズム設計実践

visual studio 2022 の使い方

演習問題

(1) cout<<4/2+(1+3)*5-15%4;

(2) cout<<((a=2)&& (b=-1));

(3)

```cpp
#include <iostream>
using namespace std;

int main()
{
    int f, f1 = 0, f2 = 1;
    for (int i = 3; i <= 6; i++)
    {
        f = f1 + f2;
        f1 = f2; f2 = f;
    }
    cout << f << endl;
    return 0;
}
```

(4)

```cpp
#include <iostream>
using namespace std;

int main() {
    int i, s = 0;
    for (i = 1; s < 20; i += 2)s += i * i;
    cout << i << endl;
    return 0;
}
```

(5) $\sum_{i=1}^{20} i$

(6) 100 以内の偶数の和。

(7) 50 以内の 3 と 5 の公倍数の和

(8) 20 以内奇数の積

7.中間テスト

8.ポインター

```
int * i;
int a;
i=&a;
*i= 5;
```

演習問題

(1) 配列｛1,7,2,5,9｝を大きい順で出力せよ。

```
#include <iostream>
using namespace std;

int main() {

    int a[5] = {1,7,2,5,9};
    int* p;
    p = a;
    for(int i =0;i<5;i++)
    {
        for (int j = i+1; j < 5; j++)
        {
            if (*(p + i) > *(p + j))
            {
                int  tmp = *(p + j);
                *(p + j) = *(p + i);
                *(p + i) = tmp ;

            }
        }
        std::cout << *(p + i)<<" ";
    }

    return 1;
}
```

9.クラス

class 名前
{
  public: //外からアクセス可、
        変数；
        void f()；
        名前();//コンストラクタ
        ~名前(); //デストラクタ
  private: //外からアクセス不可、デフォルト、そのクラス内からのみアクセス可
        変数；
        関数；
  protected: //外からアクセス不可、そのクラスと、そのクラスを継承したクラスからのみアクセス可
        変数；
        関数；
}
void 名前:: f()
{
    内容；
}

例 1：

```
#include <iostream>
using namespace std;
class Box {
public:
double length;
double width;
double height;
double get(void);
void set(double len, double bre, double hei );
};
double Box::get(void)
{ return length * width * height; }
void Box::set(double len, double wid, double hei)
 { length = len; width = wid; height = hei; }
int main( )
 {
  Box Box1;
```

```
  Box Box2;
  double volume = 0.0;
  Box1.height = 5.0;
  Box1.length = 6.0;
  Box1.width = 8.0;
  volume = Box1.height * Box1.length * Box1.width;
  cout << "Box1：" << volume <<endl;
  Box2.set(10.0, 8.0, 7.0);
  volume = Box2.get();
  cout << "Box2：" << volume <<endl;
  return 0;
}
```

例 2：

```
#include <iostream>
using namespace std;
class Line {
  public:
  void setLength( double len );
  double getLength( void );
  Line(double len); //コンストラクタ
  private:
  double length;
 };
Line::Line( double len)
 {
cout << "Object is being created, length = " << len << endl;
length = len;
}
void Line::setLength( double len )
{ length = len; }
double Line::getLength( void )
{ return length; }
int main( )
{
   Line line(10.0);
   cout << "Length of line : " << line.getLength() <<endl;
  line.setLength(6.0);
  cout << "Length of line : " << line.getLength() <<endl;
```

```
  return 0;
}
/*
Object is being created, length = 10
Length of line : 10
Length of line : 6

*/
```

例 3：

```
#include <iostream>
using namespace std;
class Line {
  public:
  void setLength( double len );
  double getLength( void );
  Line();//コンストラクタ
  ~Line(); //デストラクタ
  private:
  double length;
};
Line::Line(void)
{ cout << "Object is being created" << endl; } Line::~Line(void)
{ cout << "Object is being deleted" << endl; } void Line::setLength( double len )
{ length = len; }
double Line::getLength( void )
{ return length; }
int main( )
{
  Line line;
  line.setLength(6.0);
  cout << "Length of line : " << line.getLength() <<endl;
  return 0;
}
/*
Object is being created
Length of line : 6
Object is being deleted
*/
```

10.ロボットビジョン～画像処理～

10.1 画像とは？

10.2 色とは？　カラー画像とグレー画像

10.3 画像のサイズは？解像度

10.4 画像処理とは？

10.5 画像処理ライブラリ opencv　（opencv 4.10.0 バージョンの場合）

```
#pragma comment(lib, "opencv_world4100.lib")
// opencv_world4100.dll の追加　システム詳細設定
```

11 ロボットビジョン～画像処理演習～

11.1 ファイルから画像を読み込んで表示する

```
#include <opencv2/opencv.hpp>
#pragma comment(lib, "opencv_world4100.lib" )
int main(void)
{
	// 画像を格納するオブジェクトを宣言する
	cv::Mat	image;

	// 画像ファイルから画像データを読み込む
	image = cv::imread("C:/Program Files/opencv/sources/samples/data/lena.jpg");

	if (image.empty() == true) {
		// 画像データが読み込めなかったときは終了する
		return 0;
	}

	// ウィンドウに画像を表示する
	// # ここではウィンドウに「画像」という識別名を付けている
	cv::imshow("画像", image);

	// 何かキーが押されるまで待つ
	cv::waitKey();

	return 0;
}
```

11.2 画像をファイルに保存する

```
#include <opencv2/opencv.hpp>
#pragma comment(lib, "opencv_world4100.lib" )
int main(void)
{
	// 画像を格納するオブジェクトを宣言する
	cv::Mat	srcImage, dstImage;

	// 画像ファイルから画像データを読み込む
	srcImage = cv::imread("C:/Program Files/opencv/sources/samples/data/lena.jpg");

	if (srcImage.empty() == true) {
```

```
                // 画像データが読み込めなかったときは終了する
                return 0;
        }

        // 画像変換（画像の上下反転）
        // # srcImage を画像変換した結果を dstImage に入れる
        cv::flip(srcImage, dstImage, 0);

        // ウィンドウに画像を表示する
        cv::imshow("原画像", srcImage);
        cv::imshow("変換後画像", dstImage);

        // 画像を保存する
        // # ファイルフォーマットはファイル名の拡張子で自動判別される
        // # 保存先をフルパスで指定していないときは、ソースプログラムと同じフォルダに保存される
        cv::imwrite("変換後画像.jpg", dstImage);        // JPEG フォーマットで保存
        cv::imwrite("変換後画像.png", dstImage);        // PNG フォーマットで保存

        // 何かキーが押されるまで待つ
        cv::waitKey();

        return 0;
}
```

12. ロボットビジョン～動画処理～

12.1　動画ファイルから映像を読み込んで表示する

```
#include <opencv2/opencv.hpp>
#pragma comment(lib, "opencv_world4100.lib" )
int main(void)
{
	// 動画ファイルを取り込むためのオブジェクトを宣言する
	cv::VideoCapture cap;

	// 動画ファイルを開く
	cap.open("C:/Program Files/opencv/sources/samples/data/Megamind.avi");
	if (cap.isOpened() == false) {
		// 動画ファイルが開けなかったときは終了する
		return 0;
	}

	// 画像を格納するオブジェクトを宣言する
	cv::Mat frame;

	for (;;) {
		// cap から frame へ 1 フレームを取り込む
		cap >> frame;

		if (frame.empty() == true) {
			// 画像が読み込めなかったとき、無限ループを抜ける
			break;
		}

		// ウィンドウに画像を表示する
		cv::imshow("再生中", frame);

		// 30 ミリ秒待つ
		// # waitKey の引数にキー入力の待ち時間を指定できる（ミリ秒単位）
		// # 引数が 0 または何も書かない場合は、キー入力があるまで待ち続ける
		cv::waitKey(30);
	}
	return 0;
}
```

12.2 映像を動画ファイルに保存する

```
#include <opencv2/opencv.hpp>
#pragma comment(lib, "opencv_world4100.lib" )
int main(void)
{
	// 動画ファイルを取り込むためのオブジェクトを宣言する
	cv::VideoCapture cap;

	// 動画ファイルを開く
	cap.open("C:/Program Files/opencv/sources/samples/data/Megamind.avi");
	if (cap.isOpened() == false) {
		// 動画ファイルが開けなかったときは終了する
		return 0;
	}

	// 作成する動画ファイルの諸設定
	int    width, height, fourcc;
	double fps;

	width  = (int)cap.get(cv::CAP_PROP_FRAME_WIDTH);    // フレーム横幅を取得
	height = (int)cap.get(cv::CAP_PROP_FRAME_HEIGHT);   // フレーム縦幅を取得
	fps    = cap.get(cv::CAP_PROP_FPS);                                  // フレームレートを取得

	// ビデオフォーマットの指定
	fourcc = cv::VideoWriter::fourcc('X', 'V', 'I', 'D');          // Xvid  / ファイル拡張子  .avi
	// 参考：その他のフォーマット
//	fourcc = cv::VideoWriter::fourcc('D', 'I', 'V', 'X');          // DivX  / .avi
//	fourcc = cv::VideoWriter::fourcc('H', '2', '6', '4');          // H.264 / .wmv
//	fourcc = cv::VideoWriter::fourcc('V', 'P', '9', '0');          // VP9    / .avi
//	fourcc = cv::VideoWriter::fourcc('W', 'M', 'V', '2');          // WMV8  / .wmv
//	fourcc = cv::VideoWriter::fourcc('W', 'M', 'V', '3');          // WMV9  / .wmv
//	fourcc = cv::VideoWriter::fourcc('m', 'p', '4', 'v');          // ISO MPEG-4 / .mp4
//	fourcc = cv::VideoWriter::fourcc('M', 'P', '4', '3');          // MS MPEG-4  / .avi
//	fourcc = cv::VideoWriter::fourcc('D', 'I', 'B', ' '); // RGB 非圧縮  / .avi

	// 動画ファイルを書き出すためのオブジェクトを宣言する
	cv::VideoWriter writer;
	writer.open("ビデオ.avi", fourcc, fps, cv::Size(width, height));
```

```
	// 画像を格納するオブジェクトを宣言する
	cv::Mat frame, dst;

	for (;;) {
		// cap から frame へ 1 フレームを取り込む
		cap >> frame;

		// 画像から空のとき、無限ループを抜ける
		if (frame.empty() == true) {
			break;
		}

		// ウィンドウに画像を表示する
		cv::imshow("変換中", frame);

		// 画像処理（画像を左右反転して dst に書き込む）
		cv::flip(frame, dst, 1);

		// 画像 dst を動画ファイルへ書き出す
		writer << dst;

		// 1 ミリ秒待つ
		cv::waitKey(1);
	}

	return 0;
}
```

12.3 カメラから映像を取り込んで表示する

```
#include <opencv2/opencv.hpp>
 #pragma comment(lib, "opencv_world4100.lib" )
int main(void)
{
	// 映像を取り込むためのオブジェクトを宣言する
	cv::VideoCapture cap;

	// カメラに接続する
	// # open の引数にカメラ番号を指定する（通常は 0）
```

```
	cap.open(0);
	if (cap.isOpened() == false) {
		// カメラの接続ができなかったときは終了する
		return 0;
	}

	// 画像情報を取得
	int	width  = (int)cap.get(cv::CAP_PROP_FRAME_WIDTH);		// フレーム横幅を取得
	int	height = (int)cap.get(cv::CAP_PROP_FRAME_HEIGHT);	// フレーム縦幅を取得

	std::cout << "画像サイズ　" << width << "x" << height << std::endl;

	// 画像を格納するオブジェクトを宣言する
	cv::Mat frame;

	for (;;) {
		// 1 フレームを取り込む
		cap >> frame;

		// 画像から空のとき、無限ループを抜ける
		if (frame.empty() == true) {
			break;
		}

		// ウィンドウに画像を表示する
		cv::imshow("入力中", frame);

		// 30 ミリ秒待つ
		int key = cv::waitKey(30);

		// 何かキーが押されたら終了する
		if (key > 0) {
			break;
		}
	}

	return 0;
}
```

13. ロボットビジョン～動画演習～

（1）カメラから人の顔画像を取得し、動画として保存。

（2）パソコンに保存した顔の動画を用いて、顔検出を行う。

14.期末試験

参考資料：

[1] http://cvwww.ee.ous.ac.jp/opencv_practice1/

[2] https://www.youtube.com/watch?v=tDRsH2UzZIo

[3] https://kanagawa-u.ent.box.com/s/58551pq2uiiiv5rd67ks3gje27cpmo8r

[4] https://kanagawa-u.box.com/s/gf5y63i39m0502h3sl4g0k69cxxjczle

END