愛知障害者職業能力開発校

システム設計科

修了研究発表会報告書

題名 ： C言語の擬似クラス

あらまし： C言語でクラスを作れるという噂の真偽を確かめるために思考錯誤した。

まえがき： VC++やJavaその他オブジェクト指向の言語にはクラスが存在して、
クラスはオブジェクトの設計図である。

手法 ： C++のクラスを解析して、C++のクラスを作成して、C言語に翻訳する。
”class” → “struct” とする。thisを新規使用。

解析 ： 確かにC言語でクラスのようなものを記述することはできた。
が、本物のクラスではないのでこれを”擬似クラス”と仮に
名づけることにする。結果はC言語ではクラスを作れないものの
C言語ではクラスのようなものが作れることが分かった。

議論 ： 応用としてC言語でスレッドのようなものを作ってみた。
”擬似スレッド”と仮に名づけることにする。

図1 C++のクラスを含んだプログラム

```
//     Mnop.h
#include <stdio.h>
```

```
#include <time.h>

//     Abcd.h
namespace Abcd
{
      class CmyAbcd
      {
      protected:
            int Efgh;
      public:
            CmyAbcd();
            void setEfgh();
            int getEfgh();
            void setIjkl(int Ijkl);
      };
}

//     Abcd.cpp
#include "Mnop.h"
#include "Abcd.h"
namespace Abcd
{
      CmyAbcd::CmyAbcd()
      {
            Efgh = 0;
            setEfgh();
      }
      void CmyAbcd::setEfgh()
      {
```

```
        Efgh = clock();
        return;
    }
    int CmyAbcd::getEfgh()
    {
        return ((clock() - Efgh)/CLOCKS_PER_SEC);
    }
    void CmyAbcd::setIjkl(int Ijkl)
    {
        setEfgh();
        while(getEfgh() < Ijkl)
        {
        }
        return;
    }
}

//    Mnop_main.h
#ifndef Abcd_h
#define Abcd_h
#include "Abcd.h"
#endif

//    Mnop.cpp
#include "Mnop.h"
#include "Mnop_main.h"
using namespace Abcd;
void main(int argc, char *argv[])
{
```

```
	//	オブジェクトの宣言
	CmyAbcd a;

	printf("10秒数えます¥n");

	//	10秒待ち
	a.setIjkl(10);

	printf("10秒経ちました¥n");

	//	ポーズ
	getchar();

	//	終了
	return;
}
```

図2 C言語のクラスのようなものを含んだプログラム

```
/* Mnop.h */

/* ライブラリのインクルード */
#include <stdio.h>
#include <time.h>

/* 擬似クラスの構造体宣言 */
struct CmyAbcd
{
	int Efgh;
```

```
};

/* 擬似クラスの擬似コンストラクタのプロトタイプ宣言 */
void CmyAbcd(struct CmyAbcd *this);

/* 擬似クラスの擬似メソッドのプロトタイプ宣言 */
void setEfgh(struct CmyAbcd *this);
int getEfgh(struct CmyAbcd *this);
void setIjkl(struct CmyAbcd *this, int Ijkl);

/* Mnop.c */

/* ライブラリのインクルード */
#include "Mnop.h"

/* メイン関数 */
void main(int argc, char *argv[])
{
	/* 擬似オブジェクト(インスタンス)宣言 */
	struct CmyAbcd a;

	/* 擬似オブジェクト(インスタンス)初期化 */
	CmyAbcd(&a);

	printf("10秒数えます¥n");

	/* 10秒待ち */
	setIjkl(&a, 10);
```

```
	printf("10秒経ちました¥n");

	/* ポーズ */
	getchar();

	/* 終了 */
	return;
}

/* 擬似クラスの擬似コンストラクタ関数 */
void CmyAbcd(struct CmyAbcd *this)
{
	this->Efgh = 0;
	setEfgh(this);
}

/* 擬似クラスの擬似メソッド関数 */
void setEfgh(struct CmyAbcd *this)
{
	this->Efgh = clock();
	return;
}
int getEfgh(struct CmyAbcd *this)
{
	return ((clock() - this->Efgh)/CLOCKS_PER_SEC);
}
void setIjkl(struct CmyAbcd *this, int Ijkl)
{
```

```
        setEfgh(this);
        while(getEfgh(this) < Ijkl)
        {
        }
        return;
}
```

図3 C言語の擬似スレッド

```
/* Race.h */

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <conio.h>
#define CLEAR system("cls")
#define USE_THREAD

/* Race_Timer.h */

/* 時間に使用する定数の宣言 */
#define WOVICLOCKSIZE 1000000.0

/* 擬似スレッド定義 */
#ifdef USE_THREAD

/* 擬似スレッドに使用する定数の宣言 */
#define THREADSIZE 8
#define NULLTHREADID -1
```

```
#define INITCLOCKNO 1000001
#define STOPCLOCKNO 1000002

/* 構造体宣言 */
struct Handle_Thread
{
	/* 擬似スレッドID */
	int ID;
	/* 指定開始時 */
	double preClock;
	/* preClockがsetClock秒増えたらRunを呼ぶ */
	double setClock;
};
struct Thread
{
	struct Handle_Thread *p_ThreadID;
};

/* 擬似メソッドとwovi用関数のプロトタイプ宣言 */
/* 宣言の順番は以下の通り */
void nextRun(struct Handle_Thread *this, int ms);
void Run(struct Handle_Thread *this); /* Race_main.cで内容を定義します */
void Init(struct Handle_Thread *this); /* Race_main.cで内容を定義します */
void Thread(struct Thread *this, int id);
void Destroy(struct Thread *this);
void Start(struct Thread *this);
void Stop(struct Thread *this);
void woviRun(void);
void woviInit(void);
```

```
#endif

/* タイマ関数 */
void wovi(void);
/* タイマ初期化関数 */
void initWOVI(void);
/* ミリ秒待ち関数 */
void setSleep(unsigned int ms);

/* Race_Timer.c */

#include "Race.h"
#include "Race_Timer.h"

/* 時間を表す外部変数宣言 */
double woviClock;

/* 擬似スレッド定義 */
#ifdef USE_THREAD

/* wovi用擬似インスタンス宣言 */
struct Handle_Thread woviThread[THREADSIZE];

/* スレッドのvoid Sleep(int ms)の代用 */
void nextRun(struct Handle_Thread *this, int ms)
{
	this->preClock = woviClock;
```

```
    this->setClock = (((double) ms) / 1000);
    return;
}

/* スレッドのコンストラクタの代用 */
void Thread(struct Thread *this, int id)
{
    int i;
    for(i = 0; i < THREADSIZE; i++)
    {
        if(woviThread[i].preClock == INITCLOCKNO)
        {
            this->p_ThreadID = &woviThread[i];
            woviThread[i].ID = id;
            /* スレッドのvoid init(void)の代用 */
            Init(&woviThread[i]);
            break;
        }
    }
    return;
}

/* スレッドのデストラクタの代用 */
void Destroy(struct Thread *this)
{
    this->p_ThreadID->ID = NULLTHREADID;
    this->p_ThreadID->preClock = INITCLOCKNO;
    this->p_ThreadID->setClock = 0;
    return;
```

```
}

/* スレッドのvoid start(void)の代用 */
void Start(struct Thread *this)
{
	this->p_ThreadID->preClock = woviClock;
	return;
}

/* スレッドのvoid stop(void)の代用 */
void Stop(struct Thread *this)
{
	this->p_ThreadID->preClock = STOPCLOCKNO;
	return;
}

/* Runを呼ぶタイミング */
void woviRun(void)
{
	int i;
	for(i = 0; i < THREADSIZE; i++)
	{
		if(woviClock >= woviThread[i].preClock + woviThread[i].setClock)
		{
			/* スレッドのvoid run(void)の代用 */
			Run(&woviThread[i]);
			woviThread[i].preClock = woviClock;
		}
	}
```

```
	return;
}

/* 指定開始時OFF */
void woviInit(void)
{
	int i;
	for(i = 0; i < THREADSIZE; i++)
	{
		woviThread[i].ID = NULLTHREADID;
		woviThread[i].preClock = INITCLOCKNO;
		woviThread[i].setClock = 0;
	}
	return;
}

#endif

/* タイマ関数 */
void wovi(void)
{
	woviClock = clock() / CLOCKS_PER_SEC;
	if(woviClock >= WOVICLOCKSIZE)
	{
		printf("¥nOVERWOVI");
		woviClock = -WOVICLOCKSIZE;
	}
#ifdef USE_THREAD
	woviRun(); /* スレッドのためのRunを呼ぶタイミング */
```

```
#endif
	return;
}

/* タイマ初期化関数 */
void initWOVI(void)
{
	woviClock = -WOVICLOCKSIZE;
#ifdef USE_THREAD
	woviInit(); /* スレッドのための指定開始時OFF */
#endif
	return;
}

/* ミリ秒待ち関数 */
void setSleep(unsigned int ms)
{
	double start;
	double set;
	start = clock() / CLOCKS_PER_SEC;
	set = ((double) ms) / 1000;
	while(woviClock < start + set)
	{
		woviClock = clock() / CLOCKS_PER_SEC;
	}
	return;
}
```

```
/* Race_main.h */

#ifndef Race_Timer_h
#define Race_Timer_h
#include "Race_Timer.h"
#endif

/* Race_main.c */

#include "Race.h"
#include "Race_main.h"

#ifdef USE_THREAD
#define COURSESIZE THREADSIZE
struct Count
{
	int cnt[COURSESIZE];
};
struct Count Cnt;
#endif

void main(void)
{
	int i,j;
#ifdef USE_THREAD
	int i_cnt, j_cnt;
	/* 擬似スレッドの擬似インスタンス宣言 */
	struct Thread th[COURSESIZE];
#endif
```

```
    printf("¥nHello BCC");
    /* タイマー初期化 */
    initWOVI();
#ifdef USE_THREAD
    printf("¥nRace_Start で開始して競馬のコースが8コースありますが、");
    printf("¥n<1>コースは'r'ボタンが鞭で<2>コースは'l'ボタンが鞭です。");
    printf("¥nゴールまで80歩です。");
    /* 擬似スレッドの擬似インスタンス初期化 */
    for(i = 0; i < COURSESIZE; i++)
    {
        Thread(&th[i], (i + 1));
    }
#endif
    /* 10秒待機 */
    setSleep(10000);
#ifdef USE_THREAD
    /* 擬似スレッド開始 */
    printf("¥nThread Ready GO!¥n");
    for(i = 0; i < COURSESIZE; i++)
    {
        Start(&th[i]);
    }
    for(;;)
    {
        /* タイマー呼び出し */
        wovi();
        i_cnt = Cnt.cnt[0];
        j_cnt = 0;
        for(i = 1; i < COURSESIZE; i++)
```

```
            {
                if(i_cnt < Cnt.cnt[i])
                {
                    i_cnt = Cnt.cnt[i];
                    j_cnt = i;
                }
            }
            if(i_cnt >= 77) break;
            for(j = 0; j < 10000; j++){}
        }
        CLEAR;
        printf("GOAL!¥n<%d>WON", (j_cnt + 1));
#endif
        /* 10秒待機 */
        setSleep(10000);
        exit(0);
}

#ifdef USE_THREAD
/*
 * 擬似スレッドの擬似メソッド関数
 */
 /* public void paint(Graphics g)の代用 */
void Repaint(void)
{
        int i,j;
        CLEAR;
        for(i = 0; i < COURSESIZE; i++)
        {
```

```
            for(j = 0; j < Cnt.cnt[i]; j++)
            {
                printf(" ");
            }
            printf("<%d>", (i + 1));
            printf("¥n");
        }
        return;
}
/* スレッドのpublic void run()の代用 */
void Run(struct Handle_Thread *this)
{
        int i;
        char key = ' ';
        if(this->ID == 1)
        {
            Repaint();
            if(kbhit())
            {
                key = getche();
            }
            if(key == 'r')
            {
                Cnt.cnt[0]++;
            }
            nextRun(this, (((rand() % 9) + 10) * 30));
            while(kbhit())
            {
```

```
            key = getche();
            if(key == NULL)
            {
                break;
            }
        }
    }
    else if(this->ID == 2)
    {
        Repaint();
        if(kbhit())
        {
            key = getche();
        }
        if(key == 'l')
        {
            Cnt.cnt[1]++;
        }
        nextRun(this, (((rand() % 9) + 10) * 30));
        while(kbhit())
        {
            key = getche();
            if(key == NULL)
            {
                break;
            }
        }
    }
    for(i = 2; i < COURSESIZE; i++)
```

```
        {
            if(this->ID == (i + 1))
            {
                Repaint();
                Cnt.cnt[i]++;
                nextRun(this, (((rand() % 9) + 10) * 200));
            }
        }
        return;
    }
    /* スレッドのpublic void init()の代用 */
    void Init(struct Handle_Thread *this)
    {
        int i;
        if(this->ID == 1)
        {
            Cnt.cnt[0] = 0;
            nextRun(this, (((rand() % 9) + 10) * 30));
        }
        else if(this->ID == 2)
        {
            Cnt.cnt[1] = 0;
            nextRun(this, (((rand() % 9) + 10) * 30));
        }
        for(i = 2; i < COURSESIZE; i++)
        {
            if(this->ID == (i + 1))
            {
                Cnt.cnt[i] = 0;
```

```
                nextRun(this, (((rand() % 9) + 10) * 200));
            }
        }
        return;
}
#endif

# Race_makefile.mak
CC = bcc32
Race_main.exe : Race_Timer.obj Race_main.obj
    $(CC) Race_main.obj Race_Timer.obj
Race_Timer.obj : Race_Timer.c Race_Timer.h Race.h
    $(CC) -c Race_Timer.c
Race_main.obj : Race_main.c Race_main.h Race_Timer.h Race.h
    $(CC) -c Race_main.c
clean:
    del *.obj
    del *.tds

@rem Race_make.bat
D:
cd D:¥Electronics¥C_Race¥Race_Work
del Race_error.txt
copy Race_makefile.mak C:¥borland¥bcc55¥Bin¥Race_makefile.mak
copy Race.h C:¥borland¥bcc55¥Bin¥Race.h
copy Race_main.c C:¥borland¥bcc55¥Bin¥Race_main.c
copy Race_main.h C:¥borland¥bcc55¥Bin¥Race_main.h
copy Race_Timer.c C:¥borland¥bcc55¥Bin¥Race_Timer.c
copy Race_Timer.h C:¥borland¥bcc55¥Bin¥Race_Timer.h
```

```
C:
cd C:¥borland¥bcc55¥Bin
make -f Race_makefile.mak >> Race_error.txt
make -f Race_makefile.mak clean >> Race_error.txt
move Race_main.exe D:¥Electronics¥C_Race¥Race_Work¥Race.exe
del Race_makefile.mak
del Race.h
del Race_main.c
del Race_main.h
del Race_Timer.c
del Race_Timer.h
move Race_error.txt D:¥Electronics¥C_Race¥Race_Work¥Race_error.txt
D:
cd D:¥Electronics¥C_Race¥Race_Work
Race_error.txt
exit

MAKE Version 5.2  Copyright (c) 1987, 2000 Borland
        bcc32 -c Race_Timer.c
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
Race_Timer.c:
        bcc32 -c Race_main.c
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
Race_main.c:
        bcc32 Race_main.obj Race_Timer.obj
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
MAKE Version 5.2 Copyright (c) 1987, 2000 Borland
```

```
	del *.obj
	del *.tds
```

C言語のプロジェクト Race について、

main.c の 関数main を見てください。

スレッドを使用しています。

struct Thread th[COURSESIZE]; でオブジェクト宣言しています。
Thread(&th[i], (i + 1)); で初期値設定しています。
この2行は Java で次と同じ意味です。

```
Thread th[] = new Thread[COURSESIZE];
th[i] = new Thread(i + 1);
```

Start(&th[i]); でスレッドを開始しています。
この1行は Java で次と同じ意味です。

```
th[i].start();
```

```
void Repaint(void)
{
	...
}
void Run(struct Handle_Thread *this)
{
	...
}
void Init(struct Handle_Thread *this)
{
```

```
	...
}
```

はそれぞれ Java で次と同じ意味です。

```
public void paint(Graphics g)
{
	...
}
public void run()
{
	...
}
public void init()
{
	...
}
```

これらのスレッドに関する仕様は Timer.c に記述しました。

makefile.mak make.bat は複数のファイルを1個のプロジェクトとして
コンパイルするためのファイルです。

Race_Start で開始して競馬のコースが8コースありますが、
<1>コースは'r'ボタンが鞭で<2>コースは'l'ボタンが鞭です。
ゴールまで80歩です。