C,C++と Delphiの PASCAL言語との違い

C,C++と Delphiの PASCAL言語との違い というタイトルを付けましたが、厳密には、4つに分かれます。 Cと C++、そして PASCAL と ObjectPASCAL の 4つです。

まず、C と PASCALは、最初に出て来た基本となる言語です。 両方とも ストラクチャードプログラミングを 意識した言語で データの構造化を定義する機能があります。

Cの場合は、struct または、typedef struct で、複数の単純変数、あるいは別の 構造体変数をメンバーとする独自のデータ型を、宣言する事が出来ます。

PASCALの場合も、record 宣言で、独自のデータ型を、宣言する事が出来ます。

C言語の 構造体変数 宣言の例：

```
typedef struct  {
    int     vh;  // 上限電圧 判定値
    int     vl;  // 下限電圧 判定値
    int     ih;  // 最大電流 判定値
} MAX_VA_PARAM;  // 最大値パラメータ

MAX_VA_PARAM  Vpm;  //構造体変数宣言
```

PASCAL言語の 構造体変数 宣言の例：

```
MAX_VA_PARAM = record
    vh: integer;  // 上限電圧 判定値
    vl: integer;  // 下限電圧 判定値
    ih: integer;  // 最大電流 判定値
end;

Vpm: MAX_VA_PARAM;  //構造体変数宣言
```

このストラクチャードプログラミング、構造体変数の考え方は、のちに オブジェクト指向へと、発展していきます。 という事で、C++ と ObjectPASCAL は、オブジェクト指向言語です。 オブジェクト指向言語を正面から理解しようとすると難しい要素が、いくつか出てきます。

オブジェクト指向プログラミングの基本3概念

- カプセル化（ encapsulation ）
- 継承（ inheritance ）
- 多態性（ polymorphism ）

訳の分からないような言葉ですが、この中でカプセル化は、構造体から発生したようなところがあります。 構造体は、ある処理を行う上で、必要となる変数群を メンバー変数として ひとつに まとめた物です。

構造体に、更に そのデータ処理を行う関数群も、まとめてメンバー関数として入れ込んだ物が、オブジェクト指向でよく見る クラスです。

クラスは、ある処理を行う上で、必要なデータとデータを操作する関数群を、1本に まとめた物です。 かつ、不用意に外部から内部変数を アクセス出来ないように、プロテクトの機能も備えています。 このクラスを使う事で、重要なデータを安全に カプセル化する事が出来る訳です。 このクラスの事を オブジェクトと 呼ぶ事もあります。 次の継承は 元々あるクラスを 親クラスと呼ぶ事にします。 その親クラスから、子クラスを生成する事が出来ます。 それを 継承 と 呼びます。 ただ継承しただけであれば、全く同じ機能のクラスです。親と同じ機能ですが、親と同じコーディングは、必要ありません。 親のコードを呼び出す事が出来るのです。

このあたりから、初心者の方にとっては、難しくなってくるかもしれません。 オブジェクト指向の舞台裏というか、メモリ管理がどうなっているかというと、親の継承によるオブジェクト生成で、データ領域は新規に確保されますが、親が持っているコード（関数等のプログラム）は親の関数呼び出しの ポインタを、継承した子クラスが 持っていると思われます。よって、同じ処理の関数は、親子共通で、１本の関数を呼び出します。 で、データは、子の領域をアクセスします。 このあたりは、データ領域の先頭ポインタを持っているのでしょうね。
そして、子のクラスには、親の機能にない、新しい機能を追加する事が出来ます。
よって、子クラスでは、新しい機能だけコーディングすればいい事になります。

あと、親のメンバー関数と同じ名前で、子クラスのメンバー関数で、上書きする事も出来ます。
オーバーライドといいます。
もう一つ、同じ関数名で、引数の型や、個数が異なる関数を 複数用意する事も出来ます。
これを オーバーロードと いいます。
この場合は、呼び出し側で、引数の型がスイッチとなり、型が 一致する関数が、呼び出されます。

申し訳ありませんが、多態性は、私も よく理解していません。 状況により機能が変わるという意味では、オーバーロード、あるいは オーバーライド的な、状況を意味すると思います。

今回は、オブジェクト指向の話では無いので、このあたりで、オブジェクト指向の話は終りにします。

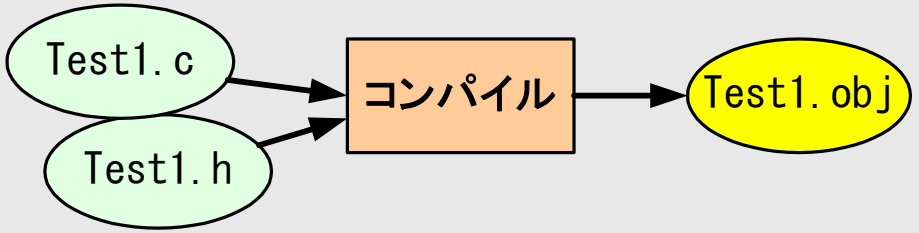
C++と Delphiの ソースファイル構成

C++の場合は、右の Test1.h内に インクルードファイルの呼び出しや、フォームの クラス宣言が作成されます。
Test1.c内に、クラス内の イベントハンドラや独自に作成した関数の実装部を 記述します。

Delphiのソースは、最初に unit名が 宣言され、その下に interface部があり、uses節で、呼び出す モジュール名を指定しています。
その下に フォームのクラス宣言が、作成されます。 その下の、var 節にて
Form1: TForm1; 左のように フォームの実態が、宣言されます。 その下に implementation 部があり、イベントハンドラの実装や、独自手続き、独自関数の 実装を 記述します。
Delphiの場合は、C言語の ヘッダファイルと Cファイルが、ひとつになったような状態です。

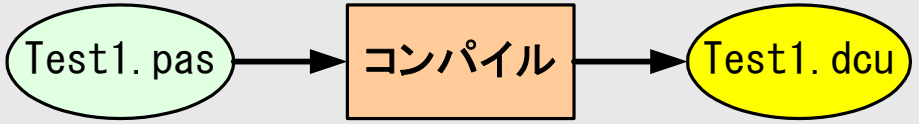
C++言語のソースファイル構成： Test1の場合

1つのフォームに Test1.Cと Test1.H のソースが生成される。コンパイルで Test1.objが生成される。



Delphiの ソースファイル構成：Test1の場合

1つのフォームに Test1.pasのソースが 生成される。コンパイルで Test1.dcuが 生成される。



C、C++と Delphiの 違い

項目	C、C++	Delphi
変数の宣言 （表現できる レンジ） 変数のByte数	char b; (-128～127) 1 byte short int i; (-32768～32767) 2 byte int i; (-2147483648～2147483647) 4 byte long int i; (-2147483648～2147483647)4 byte unsigned char b; (0～255) 1 byte unsigned short int i; (0～65535) 2 byte unsigned int i; (0～4294967295) 4 byte unsigned long int i;(0～4294967295)4 byte float a; (1.17E-38～3.40E+38) 4 byte double a; (2.22E-308～1.79E+308) 8 byte long double a; (2.22E-308～1.79E+308)8 byte 参照したサイトの情報が古い可能性があります	b: ShortInt; (-128～127) 1 byte i: SmallInt; (-32768～32767) 2 byte i: integer; (-2147483648～2147483647) 4 byte i: Int64; (-2^{63}～$2^{63}-1$) 8 byte b: Byte; (0～255) 1 byte w: Word; (0～65535) 2 byte i: Cardinal; (0～4294967295) 4 byte i: UInt64; (0～$2^{64}-1$) 8 byte a: Single; (1.17E-38～3.40E+38) 4 byte a: Double; (2.22E-308～1.79E+308) 8 byte a: Extended; (？～？) 10 byte

基本的なところでの違いは、記述の仕方で C、C++は 変数型名 変数名; ですが、 Delphiは、左右逆で、変数名: 変数型名; です。

Delphiの 10byte 浮動小数点の Extended は、Delphi 10.4 Community Edition で、使えるのか試したところ 使えました。sizeof 演算子で サイズを確認したところ 10byte でした。レンジが どの程度あるのかは不明です。

少なくとも Doubleより高精度と 思います。

項目	C、C++	Delphi

関数と手続き	C,C++では 値を返す、返さないに関わらず 関数として扱います。
	intの値を返す関数：
	int test_func(int a);
	値を 返さない関数：
	void test_func(int a);
	引数を渡さない関数：
	int test_func(void);
		Delphiでは、値を返さない処理を 手続き
		値を返す処理を 関数として扱います。
		Integerの値を返す関数：
		function test_func(a: Byte): Integer;
		値を返さない手続き：
		procedure test_proc(a: Byte);
		引数を渡さない手続き：
		procedure test_proc;
		引数を囲むカッコから、省略されます。

文字列変数宣言	通常は、以下の表現で使えます。
	String tx; // 文字列変数の宣言
	または AnsiString tx;
	tx = "Abcd"; // 文字列の代入
	昔の char tx[80]; とかの Null終端の文字列も 使用できます。
		Delphiの文字列は、以下の表現で使えます。
		tx: String; // 文字列変数の宣言
		tx := 'Abcd'; // 文字列の代入
		ASCII文字列 255文字までの 短い文字列変数もあります。 ←― この部分が最大255です。
		tx: String[80]; // この場合は、最大
		// 80文字までの文字列変数になります。

項目	C、C++	Delphi

値の代入	int j; に 10を 代入する時
	j = 10; // = で 右辺を 左辺に
	// 代入します。
		j: Integer; に 10を 代入する時
		j := 10; // := で 右辺を 左辺に
		// 代入します。

if 文で使用する 関係演算子	== は、左辺と右辺が等しい時 True != は、左辺と右辺が等しくない時 True > は、左辺が 右辺より大きい時 True < は、右辺が 左辺より大きい時 True >= は、左辺が 右辺より大きいか、等しい時 True <= は、右辺が 左辺より大きいか、等しい時 True	= は、左辺と右辺が等しい時 True <> は、左辺と右辺が等しくない時 True > は、左辺が 右辺より大きい時 True < は、右辺が 左辺より大きい時 True >= は、左辺が 右辺より大きいか、等しい時 True <= は、右辺が 左辺より大きいか、等しい時 True

文字列定数の
囲み
	″～″を 使用する。
	tx = ″ABCDE″;
	c = 'A'; // 1byteの文字変数に 文字
	// コードの代入は 'を使う
		1文字でも 文字列でも ' を使用する。
		tx := 'ABCDE';
		c := Ord('A'); // byte変数 c に Aの
		// 文字コードを 代入

Edit1の Textプロパティを示す	Edit1->text = Edit2->Text;	Edit1.Text := Edit2.Text;

項目	C、C++	Delphi

複数の文を囲む	中カッコを { ～ }を 使います。
	if(sw == 1)
	{ sw = 1; j = 2; }
	else
	{
	sw = 0;
	j = 3;
	}
		begin ～ end を 使います。
		if sw = 1 then
		begin
		sw := 1; j := 2;
		end ← // 後ろに elseが ある時は ;を
		else // 付けては いけない。
		begin
		sw := 0:
		j := 3;
		end;

if ～ else 文	if(関係式) True時の実行文
	else False時の実行分
	if(sw == 1)
	sw = 0;
	else
	sw = 1;
		if 関係式 then True時の実行分
		else False時の実行分
		if sw = 1 then
		sw := 0 ←// 後ろに elseが ある時は
		else // ;を 付けてはいけない
		sw := 1;

項目	C、C++	Delphi

for文	for(i=0; i<10; i++)
	{ ～ }
	for(i=10; i>0; i--)
	{ ～ }
		for i:=0 to 9 do
		begin ～ end;
		for i:=10 downto 1 do
		begin ～ end;
		増分 減分を 2とか 3の指定は 出来ません

while文
break;
continue;
	while(ループする条件式)
	{
	// 処理A
	if(ループ中断の条件式) break;
	// 処理B
	if(先頭に戻る条件式) continue;
	// 処理C
	}
		while ループする条件式 do
		begin
		// 処理A
		if ループ中断の条件式 then break;
		// 処理B
		if 先頭に戻る条件式 then continue;
		// 処理C
		end;

大文字と小文字の区別	C 及び C++ は、大文字と小文字を別の文字として区別します。 Abc と abc は、異なる変数になります。	Delphiは、変数名などの 大文字と小文字の違いを 区別しません。 Abc と abc は、同じ変数になります。

項目	C、C++	Delphi

switch
case
default
 	
case of
else
	switch(式)
	{
	  case 1: a = 100; // 式=1 の時
	  break; // 中断
	  case 2: a = 200; // 式=2 の時
	  b = 500;
	  break; // 中断
	  default: // 式が それ以外
	  a = 50; // の場合
	  b = 60;
	}
		case 式 of
		  1: a := 100; // 式=1 の時
		  2: begin //文が２つ以上ある場合
		  a := 200; // 式=2 の時
		  b := 500;
		  end
		  else begin // 式が それ以外
		  a := 50; // の場合
		  b := 60;
		  end;
		end;
		break; が いらない代わりに 文が２つ以上ある場合は、begin ～ end; で挟む事。

コメントの扱い	/* ～ */ で、囲まれた範囲 // から行末まで	{ ～ } で、囲まれた範囲 (* ～ *) で、囲まれた範囲 // から行末まで

項目	C、C++	Delphi

変数の
宣言場所
	関数内の先頭で宣言する場合
	void __fastcall TForm1::test1(void)
	{
	AnsiString tx; // 関数内でのみ
	～～～ // 有効な変数
	}
	呼び出された時、Stack上に変数エリアを確保される Auto変数になります。
	関数から呼び出し元に戻る時、変数エリアは、廃棄されます。
	---------- 	
	処理関数を含むクラス内で変数を 宣言する場合は、private: 内に宣言すると、外からアクセス出来ない変数になります。
	public: 内に宣言すると、外からアクセス出来る変数になります。
	クラスが、存在する間は、スタティックに値を保持します。
		関数内の先頭で宣言する場合
		procedure TForm1.test1;
		var // 変数宣言する時は Varが必要
		tx: String; //関数内でのみ有効な変数
		begin
		end;
		呼び出された時、Stack上に変数エリアを確保される Auto変数になります。
		関数から呼び出し元に戻る時、変数エリアは、廃棄されます。
		---------- 	
		処理関数を含むクラス内で 変数を宣言する場合は、private: 内に宣言すると、外からアクセス出来ない変数になります。
		public: 内に宣言すると、外からアクセス出来る変数になります。
		クラスが、存在する間は、スタティックに値を保持します。

項目	C、C++	Delphi
算術演算子	加算：＋、減算：－、乗算：*、除算：/ 整数型のあまり：% C特有のコーディング量を減らす演算子 a += b;　//　a = a + b; a -= b;　//　a = a - b; a *= b;　//　a = a * b; a /= b;　//　a = a / b; +1を 行う演算子 a++; または ++a; -1を 行う演算子 a--; または --a;	加算：＋、減算：－、乗算：*、 整数型の除算：DIV、実数型の徐算：/ 整数型の余り：MOD コーディング量を減らす演算子は、無いのですが、+1 を 行う処理で inc(a); -1 を 行う処理で dec(a); が あります。 これらは、アセンブラでいうところの inc命令、dec命令に置き換えられるそうです。 速度面では、多少メリットが、あるかもしれません。

論理関係
演算子
&& ||
AND OR
	if文内の関係演算子で、例えば
	if((a > b) && (c > d))の 場合
	a は b より大きく、かつ c は d より大きい場合 if文は True に なります。
	if((a > b) || (c > d))の場合
	aは bより大、 cは dより大の 片方を満たせば if文は True に なります。
		左の関係を Delphiで 表すと
		if (a > b) AND (c > d) then の場合
		a は b より大きく、かつ c は d より大きい場合 if文は True に なります。
		if (a > b) OR (c > d) then の場合
		aは bより大、 cは dより大の 片方を満たせば if文は True に なります。

項目	C、C++	Delphi

ビット
演算子
|、&、~、^
AND、OR、
NOT、XOR
	その前に 16進数の定数の表示例：0x40
	と表します。
	変数上の特定のビットを操作する時、
	
	b3を 1 にする場合
	b = b | 0x08;
	
	b3を 0 にする場合
	b = b & 0xF7;
	
	変数の ビット反転を行う
	b = ~b;
	
	2つの値の 食い違ったビットを 1にする
	c = a ^ b;
		その前に 16進数の定数の表示例：$40
		と表します。
		変数上の特定のビットを操作する時、
		
		b3を 1 にする場合
		b := b OR $08;
		
		b3を 0 にする場合
		b := b AND $F7;
		
		変数の ビット反転を行う
		b := NOT b;
		
		2つの値の 食い違ったビットを 1にする
		c := a XOR b;

ビットシフト 演算子 <<、>> SHL、SHR	左シフトの例： b = b << 2; 右シフトの例： b = b >> 3;	左シフトの例： b := b SHL 2; 右シフトの例： b := b SHR 3;

項目	C、C++	Delphi

ポインターと参照渡し	C++になってから、参照渡しが、出来るようになりました。 Cで、ポインターを使いこなしている方は、C++になってもポインターを 使っている方が 多い気がします。
	int buf[100], j;
	int *ptr;
	ptr = &buf;
	j = *ptr; //バッファの先頭データが
	// j に 入る
	ptr++; // 結果 2番目のデータを
	// 指している
	----- 参照渡し ----------
	short num = 50; // 変数定義
	short & refnum = num; // 参照変数の
	// 定義
	refnum = 80; // num=80を、
	// 行ったのと同じになる
		Delphiでは、最初から参照渡しを使っていたので、私は、Delphiでは、あまりポインターは 使いません。
		var
		buf: array [0..99] of Integer;
		j: Integer;
		ptr: ^Integer;
		begin
		ptr := @buf;
		j := ptr^;
		inc(ptr);
		～～～
		end;
		Delphiの参照渡し／同じ例が作れないので手続きの引数に参照渡しを行う例を示します。（ この使い方が 多いと思います ）
		procedure MyProc(var x: Integer);
		x := x + 10; // 呼び出し元の 変数に
		// +10が 反映される