

From PASCAL to C/C++

Chan Siu Man
siuman@hkoi.org

Hong Kong Olympiad in Informatics

January 7, 2006



Agenda

1 Overview

- Similarities
- Differences

2 Basics

- Numeric Data Types
- Hello World!
- Declaration of Variables
- Operators

3 Constructs

- IF Constructs
- WHILE Constructs
- FOR Constructs

- CASE Constructs

4 Procedures and Functions

- Procedures
- Functions
- Parameters

5 Packed Data Types

- Arrays
- Strings
- User Data Types

6 Input/Output

- IO Routines
- IO Formatting

7 Conclusion



Similarities

Both are:

- strongly influenced by ALGOL 60;
- compiled languages;
- procedural languages;
- implementing structured programming concepts; and
- equipped with functionality for dynamic allocation of memory.



Differences

Pascal

Designed as a teaching language.

Syntax are easier to understand.

Error-prone constructs are carefully avoided.

Case insensitive.

More will be discussed.

C/C++

Designed for professional programmers.

Syntax are harder to understand.

Constructs emphasize brevity and orthogonality.

Case sensitive.



Numeric Data Types

Pascal

Specifies the range.

Boolean

Char

Byte

Word

Integer

LongInt

Single

Double

C/C++¹

Specifies the length.

bool

unsigned char

signed char

unsigned short

int

long

float

double

¹The above correspondence is not exact.

Hello World!

Pascal

```
Program HelloWorld;  
Begin  
  WriteLn('Hello World!')  
End.
```

Program body must be named “main” in C/C++, and must return `int` as exit status (0 denoting successful).

“`#include <stdio.h>`” is like “Uses Crt;”.

C/C++

```
#include <stdio.h>  
int main() {  
  printf("Hello World!\n");  
  return 0;  
}
```



Declaration of Variables

Pascal

```
Program Variables;  
Var  
  i, j, k: Integer;  
  c      : Char;  
Begin  
  (* Some codes here *)  
End.
```

Identifiers are followed by types.
Declaration in Var block.

C/C++

```
int main() {  
  int i, j, k;  
  char c;  
  /* some codes here */  
  return 0;  
}
```

Types are followed by identifiers.
Declaration must precede codes
in C.



Arithmetic and Assignment Operators

Operation	Pascal	C/C++
Addition	+	+
Subtraction	-	-
Multiplication	*	*
Division	div, /	/
Modulo	mod	%
Increment	Inc(X)	++ x, x ++ ²
Decrement	Dec(X)	-- x, x -- ²
Assignment	:=	=

²Use with care!

Relational Operators

Operation	Pascal	C/C++
Equal	=	==
Not Equal	<>	!=
Greater Than	>	>
Less Than	<	<
Greater Than or Equal	>=	>=
Less Than or Equal	<=	<=



Logical/Bitwise Operators

Operation	Pascal	C/C++
Logical AND	And	&&
Logical OR	Or	
Logical NOT	Not	!
Logical XOR	Xor	^
Bitwise AND	And	&
Bitwise OR	Or	
Bitwise NOT	Not	~
Bitwise XOR	Xor	^
Shift Left	ShL	<<
Shift Right	ShR	>>



Miscellaneous Operators

Operation	Pascal	C/C++
Size of	SizeOf()	sizeof()
Address of	@	&
Dereference/Pointer to	^	*
Member selection	.	., -> (pointer)
Type casting	<i>Type (Expression)</i>	<i>(Type) Expression</i>



IF Constructs

Pascal

```
If 1 > 0 Then
  WriteLn('1>0!')
Else
Begin
  (* Multi-statement here *)
End;
```

Use `Begin` and `End` to quote multi-statement codes.
`Else` block can be omitted.
Parenthesis for condition checking can be omitted.

C/C++

```
if (1 > 0)
  printf("1>0!\n");
else {
  /* multi-statement here */
}
```

Use “{” and “}” to quote multi-statement codes.
`else` block can be omitted.
Parenthesis for condition checking must be used.



WHILE Constructs

Pascal

```
While 2 + 2 = 5 Do  
Begin  
  (* Codes here *)  
  Break;  
End
```

Use **Break**, **Continue** for additional flow control.

C/C++

```
while (2 + 2 == 5) {  
  /* Codes here */  
  break;  
}
```

Use **break**, **continue** for additional flow control.



FOR Constructs

Pascal

```
For i := 1 To N Do  
Begin  
    (* Codes here *)  
End;
```

Auto increment in the loop.
Use **DownTo** for looping backwards.

C/C++

```
for (i = 1; i <= n; ++ i) {  
    /* codes here */  
}
```

Manual increment in the header.
Use decrement (**-- i**) for looping backwards.



CASE Constructs

Pascal

Case X Of

```
1: WriteLn('X is 1.');
```

```
2: WriteLn('X is 2.');
```

```
Else WriteLn('Error');
```

End;

Only one case will be executed.
No Break is needed to execute only one case.

C/C++

```
switch (x) {
```

```
  case 1: printf("X is 1.\n");
```

```
          break;
```

```
  case 2: printf("X is 2.\n");
```

```
          break;
```

```
  default: printf("Error\n");
```

```
}
```

Like `goto`, subsequent cases will be executed. `break` is needed to execute only one case.



Procedures

Pascal

```
Procedure P1(N, M: Integer);  
Var  
  i, j: Integer;  
Begin  
  (* Some codes here *)  
End;
```

Nothing is returned.

C/C++

```
void p1(int n, int m) {  
  int i, j;  
  /* some codes here */  
}
```

void is the return type.



Functions

Pascal

```
Function F1(N: Char): Word;  
Begin  
  (* Some codes here *)  
  F1 := 1  
  (* More codes here *)  
End.
```

Return value with assignment.
Assignment does not halt
subroutine.

C/C++

```
short f1(char n) {  
  short ret;  
  /* Some codes here */  
  ret = 1;  
  /* More codes here */  
  return ret;  
}
```

Return value with **return**.
return halts subroutine.



Parameters

Pascal

Parameter types can be collectively specified. e.g.

Procedure P1(A, B: Integer);

Parenthesis must be omitted if no parameters. e.g. **Procedure P1;**

Arguments can be passed by value or by reference with the help of **Var** keyword.

C/C++

Parameter types must be listed one by one. e.g.

void p1(int a, int b)

Parenthesis cannot be omitted.

e.g. **void p1()**

Arguments must be passed by value. To pass variables by reference, the corresponding pointers are passed instead.



Arrays

Pascal

```
Var
  A: Array[1..10] of Integer;
  B: Array[-10..10] of Char;
Begin
End.
```

Arbitrary range.

Range specification is inclusive - inclusive.

Arrays are arrays.

C/C++

```
int main() {
  int a[10];
  char b[21];
  return 0;
}
```

Zero-based.

Range specification is inclusive - exclusive.

Arrays are pointers.



Strings

Pascal

```
Var
  S: String;
Begin
  S := 'HKOI';
End.
```

Single-quoted string literals.
Built-in **String** data types.

C/C++

```
#include <string.h>
int main() {
  char s[255];
  strcpy(s, "HKOI");
  return 0;
}
```

Double-quoted string literals.
NULL-terminating character
array.



User Data Types

Pascal

Type

```
Student = Record
  Name: String;
  ID  : Integer;
End;
```

Declare user data types in Type block.

C/C++

```
typedef struct student {
  char name[255];
  int id;
} student;
```

Declare user data types with typedef keywords.



IO Routines

Pascal

```
Var
  I: Integer;
  C: Char;
Begin
  ReadLn(I, C);
  WriteLn(C, ' ', I);
End.
```

C/C++

```
#include <stdio.h>
int main() {
  int i;
  char c;
  scanf("%d%c\n", &i, &c);
  printf("%c %d\n", c, i);
}
```

“#include <stdio.h>” is needed.



Pascal

`ReadLn` and `WriteLn` take arguments of any number of different types automatically.

Arguments are passed by reference automatically with `ReadLn`.

C/C++

`scanf` and `printf` take arguments of any number of different types with the help of *format string*, which specifies the order and type of each argument.

Arguments need to be passed by reference manually with `scanf`, i.e. pass the pointers by value instead. Operator `&` is used to get the addresses of the variables.

IO Formatting

Common format string tokens:

- `%d` for `int`.
- `%c` for `char`.
- `%ld` for `long`.
- `%lf` for `double`.
- `%s` for `NULL`-terminating strings.

Pay special attention to whitespace handling with `scanf!`

Escape sequences: (applicable to any string literals)

- `\n` for new line.
- `\"` for character `"`.
- `\\` for character `\`.



Recall:

- NULL-terminating strings are character arrays.
- Arrays are pointers.
- Passing by reference is passing the pointers by value.

⇒ We don't need to get the address of NULL-terminating strings with `scanf`. e.g.

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

```
int main() {  
    char s[255];  
  
    scanf("%s", s);  
    printf("%s\n", s);  
    return 0;  
}
```



Conclusion

- The syntax taught is the most generic C/C++ syntax for compatibility concern. The same idea may be expressed more easily and briefly if you are using C++ only syntax.
- Today's talk is by no means complete. There are many delicate points and programming tricks in C/C++. Do Google search!
- Try to translate some of your PASCAL programs into C/C++ now!

