The C++ Programming Language

An Example of C++ to C Translation

Outline

Introduction

C++ Source Code

C Source Code

Introduction

- The following slides illustrate the result of compiling C++ code into C code using the AT&T cfront translator.

- The example illustrates the use of data abstraction, single inheritance, and dynamic binding.

- Note that the C source code is over twice as long as the C++ source code...

C++ Source Code

- C++ Source Code

```
extern "C" { extern int printf (char *, ...); }

enum {
   ZERO, ONE, TEN = 10,
   HUNDRED = 100, THOUSAND = 1000
};

class Base {
private:
   int B_private_Data;
protected:
   int B_protected_Data;
public:
   int B_public_Data;

   Base (int arg = ONE) : B_private_Data (arg + THOUSAND),
                          B_protected_Data (arg + HUNDRED),
                          B_public_Data (arg)  {}
   ~Base (void) {}

   int B_getPrivate_Data (void) { return B_private_Data; }

   virtual int F1 (void) = 0;
   virtual int F2 (void) { return B_private_Data; }
};
```

C++ Source Code (cont'd)

- *e.g.*,

```
class Derived : public Base {
private:
   int D_private_Data;
public:
   int D_public_Data;
   Derived (int arg = ZERO) : D_private_Data (arg),
                              D_public_Data (arg),
                              Base (arg) {}
   ~Derived (void) {}

   int D_getPrivate_Data (void) { return D_private_Data; }

   virtual int F1 (void) {
      return B_protected_Data;
   }

   virtual int F2 (void) {
      return B_protected_Data + TEN + ONE;
   }
   /*
      Derived inherits:
      B_get_private_data, B_public_data, B_protected_data
   */
};
```

C++ Source Code (cont'd)

- *e.g.*,

```
int main (void) {
   Base    *bp = new Derived; // defaults to 0
   Derived d (2);

   printf ("%d\n", d.D_public_Data ); // 2
   d.D_public_Data = TEN;
   printf ("%d\n", d.D_public_Data); // 10

   printf ("%d\n", bp->B_getPrivate_Data ()); // 1000

   printf ("%d\n", d.D_getPrivate_Data ()); // 2

   /* Elide virtual function call! */
   printf ("%d\n", d.F1 ()); // 102

   /* Virtual function call! */
   printf ("%d\n", bp->F2 ()); // 111
   delete bp;
   /* Derived d destructor implicitly called. */
   return 0;
}
```

C Source Code

- Cfront output

```
/* Dynamic memory functions */
extern void *__vec_new (void *, int, int, void *);
extern void __vec_delete (void *, int, int, void *, int, int);
extern void *__nw__FUi (unsigned int);
extern void __dl__FPv (void *);

extern int printf (char *,...);

/* Type decls for multiple and single inheritance. */
typedef int (*__vptp) ();

struct __mptr {
   short d; short i; __vptp f;
};

extern struct __mptr *__ptbl_vec__c___src_C_[];

enum __E1 {
   ZERO = 0, ONE = 1, TEN = 10,
   HUNDRED = 100, THOUSAND = 1000
};
```

C Source Code (cont'd)

- *e.g.*,

```
struct Base { /* sizeof Base == 16 */
   int B_private_Data__4Base;
   int B_protected_Data__4Base;
   int B_public_Data__4Base;
   struct __mptr *__vptr__4Base; /* Compiler generated. */
};

/* Base class constructor */
static struct Base *
__ct__4BaseFi (struct Base *__0this, int __0arg) {
   /* The following 5 lines are the compiler-generated preamble. */
   if (__0this ||
      (__0this =
         (struct Base *) __nw__FUi (sizeof (struct Base))))
            ((((__0this->__vptr__4Base =
               (struct __mptr *) __ptbl_vec__c___src_C_[0]),
            /* User constructor code starts here! */
            (__0this->B_private_Data__4Base =
               (__0arg + 1000))),
            (__0this->B_protected_Data__4Base =
               (__0arg + 100))),
            (__0this->B_public_Data__4Base =
               __0arg))
      ;
   return __0this;
}
```

C Source Code (cont'd)

- *e.g.*,

```
/* Base class destructor */
static char
__dt__4BaseFv (struct Base *__0this, int __0__free) {
   if (__0this) {
      __0this->__vptr__4Base =
         (struct __mptr *) __ptbl_vec__c___src_C_[0];
      /* User destructor code would go here. */
      if (__0this)
         if (__0__free & 1)
            __dl__FPv ((char *) __0this);
   }
}

static int
B_getPrivate_Data__4BaseFv (struct Base *__0this) {
   return __0this->B_private_Data__4Base;
}

static int
F2__4BaseFv (struct Base *__0this) {
   return __0this->B_private_Data__4Base;
}
```

C Source Code (cont'd)

- *e.g.*,

```
struct Derived { /* sizeof Derived == 24 */
   int B_private_Data__4Base;
   int B_protected_Data__4Base;
   int B_public_Data__4Base;
   struct __mptr *__vptr__4Base;
   int D_private_Data__7Derived;
   int D_public_Data__7Derived;
};

/* Derived class constructor */
static struct Derived *
__ct__7DerivedFi (struct Derived *__0this, int __0arg) {
   /* The following 7 lines are the compiler-generated preamble. */
   if (__0this ||
      (__0this = (struct Derived *)
         __nw__FUi (sizeof (struct Derived))))
         ((((__0this = (struct Derived *)
            __ct__4BaseFi (((struct Base *) __0this), __0arg)),
         (__0this->__vptr__4Base =
            (struct __mptr *) __ptbl_vec__c___src_C_[1])),
         /* User constructor code goes here. */
         (__0this->D_private_Data__7Derived = __0arg)),
         (__0this->D_public_Data__7Derived = __0arg))
      ;
   return __0this;
}
```

C Source Code (cont'd)

- *e.g.*,

```
/* Derived class destructor */
static char
__dt__7DerivedFv (struct Derived *__0this, int __0__free) {
   if (__0this) {
      __0this->__vptr__4Base =
         (struct __mptr *) __ptbl_vec__c___src_C_[1];
      /* User destructor code goes here. */
      if (__0this) {
         __dt__4BaseFv (((struct Base *) __0this), (int) 0);
             if (__0__free & 1)
               __dl__FPv ((char *) __0this);
      }
   }
}

static int
D_getPrivate_Data__7DerivedFv (struct Derived *__0this) {
   return __0this->D_private_Data__7Derived;
}

static int
F1__7DerivedFv (struct Derived *__0this) {
   return __0this->B_protected_Data__4Base;
}

static int
F2__7DerivedFv (struct Derived *__0this) {
   return ((__0this->B_protected_Data__4Base + 10) + 1);
}
```

C Source Code (cont'd)

- *e.g.*,

```
int main (void) {
   _main ();
   {
      struct Base *__1bp; struct Derived __1d;

      __1bp = (struct Base *)
         __ct__7DerivedFi ((struct Derived *) 0, (int) 0);
      __ct__7DerivedFi (&__1d, 2);

      printf ((char *) "%d\n", __1d.D_public_Data__7Derived);

      __1d.D_public_Data__7Derived = 10;
      printf ("%d\n", __1d.D_public_Data__7Derived);

      printf ("%d\n", B_getPrivate_Data__4BaseFv (__1bp));

      printf ("%d\n", D_getPrivate_Data__7DerivedFv (&__1d));

      printf ("%d\n", F1__7DerivedFv (&__1d));

      /* Virtual function call */
      printf ((char *) "%d\n",
         ((*(((int (*) ()) (__1bp->__vptr__4Base[2]).f))))
            (((struct Base *) ((((char *) __1bp))
            + (__1bp->__vptr__4Base[2]).d))));

      __dt__4BaseFv (__1bp, 3);
      __dt__7DerivedFv (&__1d, 2);
      return 0;
   }
```

```
}
```

C Source Code (cont'd)

- *e.g.*,

```
struct __mptr __vtbl__7Derived__c___src_C[] = {
   0, 0, 0,
   0, 0, (__vptp) F1__7DerivedFv,
   0, 0, (__vptp) F2__7DerivedFv,
   0, 0, 0
};

char __pure_virtual_called ();

struct __mptr __vtbl__4Base__c___src_C[] = {
     0, 0, 0,
     0, 0, (__vptp) __pure_virtual_called,
     0, 0, (__vptp) F2__4BaseFv,
     0, 0, 0
};

struct __mptr *__ptbl_vec__c___src_C_[] = {
   __vtbl__4Base__c___src_C,
   __vtbl__7Derived__c___src_C,
};
```