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

```cpp
extern "C" { extern int printf (char *, ...); }
enum {
    ZER0, 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.,

```cpp
class Derived : public Base {
    private:
        int D_private_Data;
    public:
        int D_public_Data;
        Derived (int arg = ZER0) : 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.,

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

  printf ("%4d\n", d.d_public_data); // 2
  d.d_public_data = TEN;
  printf ("%4d\n", d.d_public_data); // 10
  printf ("%4d\n", bp->getPrivate_Data ()); // 1000
  printf ("%4d\n", d.d_getPrivate_Data ()); // 2

  /* Elide virtual function call */
  printf ("%4d\n", d.F1()); // 102

  /* Virtual function call */
  printf ("%4d\n", bp->F2()); // 111
  delete bp;
  /* Derived d destructor implicitly called. */
  return 0;
}
```

C Source Code

- Cfront output

```cpp
/* Dynamic memory functions */
extern void *_vec_new (void *, int, int, void *);
extern void *_vec_delete (void *, int, int, void *, int, int);
extern void *-_vec_FPi (assigned int);
extern void _-_FPi (void *);
extern int printf (char *, ...);

struct _-_mptr {
  short d; short i; _-_vpt p;
};

extern struct _-_mptr _-_ptbl__vec__src__C[ ];
enum _-_E1 {
  ZERO = 0, ONE = 1, TEN = 10,
  HUNDRED = 100, THOUSAND = 1000
};
```

C Source Code (cont’d)

- e.g.,

```cpp
struct Base { /* sizeof Base == 16 */
  int B_private_Data_4Base;
  int B_protected_Data_4Base;
  struct _-_mptr _-_vptr__4Base; /* Compiler generated. */
};

/* Base class constructor */
static struct Base * __ct__4BaseF1 (struct Base * _-_this, int _-_arg) {
  /* The following 5 lines are the compiler-generated preamble. */
  if (_-_this != NULL) {
    _-_this =
      (struct Base *) _-_vec_FPi (sizeof (struct Base));
  } else { 
    ((-_this->-_vptr__4Base = (struct _-_mptr * )-_ptbl__vec__src__C[0]),
     /* User constructor code starts here! */
     (_-_this->B_private_Data_4Base = (_-_arg + 1000)));
    (_-_this->B_protected_Data_4Base = (_-_arg + 100)),
    (_-_this->B_public_Data_4Base = _-_arg))
  
  return _-_this;
}
```

```
```
C Source Code (cont'd)

- e.g.,

```
struct Derived /* sizesof Derived = 24 */
  int B_private_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 /* Derived */
  __ct__DerivedFv (struct Derived * _this, int _0arg)
  { /* The following 7 lines are the compiler-generated preamble. */
    if (_this) { /* Derived */
      _this = (struct Derived *)
        __sy__Psi (sizeof (struct Derived));
      ((int)_this = (struct Derived *)
        __ct__DerivedFv ((struct Derived *) _this), _0arg);
      ((int)_this-> _vptr_4Base =
        (struct _mptr *) _publ_rec_.c_src.C[1]),
      /* User constructor code goes here. */
      (_this->B_private_Data_7Derived = _0arg),
      (_this->B_public_Data_7Derived = _0arg));
      return _this;
    }
  }
```

C Source Code (cont'd)

- e.g.,

```
int main (void) {
  _main ();
  { struct Base * _1bp; struct Derived _1id;
    _1bp = (struct Base *)
      __ct__DerivedFv ((struct Derived *) 0, (int) 0);
    __ct__DerivedFv (_1id, 2);
    printf ((char *) "%An", _1id.B_public_Data_7Derived);
    _1id.B_public_Data_7Derived = 10;
    printf ((char *) "%An", _1id.B_public_Data_7Derived);
    printf ((char *) "%An", B_getPrivate_Data_7DerivedFv (_1bp));
    printf ((char *) "%An", B_getPrivate_Data_4BaseFv (_1bp));
    printf ((char *) "%An", _1id.B_public_Data_7DerivedFv (_1id));
    printf ((char *) "%An", _1id.B_public_Data_7DerivedFv (_1id));
    /* Virtual function call */
    printf ((char *) "%An",
      /*(*(*(_1bp->_vptr_4Base[2].f))))
        (((struct Base *) (_1bp->_vptr_4Base[2].f))
      * (_1bp->_vptr_4Base[2].d)));
    __ct__DerivedFv (_1bp, 3);
    __ct__DerivedFv (_1id, 2);
    return 0;
  }
```
C Source Code (cont’d)

- e.g.,

```c
struct __mptr __vtbl__Derived_c__src_C[] = {
  0, 0, 0,
  0, 0, (__vptp) P1_DerivedPv,
  0, 0, (__vptp) P2_DerivedPv,
  0, 0, 0
};
char __pure_virtual_called();
```

```c
struct __mptr __vtbl__Base_c__src_C[] = {
  0, 0, 0,
  0, 0, (__vptp) __pure_virtual_called,
  0, 0, (__vptp) P2_BasePv,
  0, 0, 0
};
```

```c
struct __mptr *__vtbl__vec_c__src_C[] = {
  __vtbl__Base_c__src_C,
  __vtbl__Derived_c__src_C,
};
```