



Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing
C Solution

Summary

Module 29: Programming in C++

Dynamic Binding (Polymorphism): Part 4

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan



Module Objectives

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing
C Solution

Summary

- Understand design with class hierarchy



Module Outline

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing
C Solution

Summary

- Binding Exercise
- Staff Salary Processing
 - C Solution
 - C++ Solution
 - Non-Polymorphic Hierarchy
 - Polymorphic Hierarchy
 - Polymorphic Hierarchy (Flexible)



Binding: Exercise

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing
C Solution

Summary

```
// Class Definitions
class A { public:
    virtual void f(int) { }
    virtual void g(double) { }
    int h(A *) { }
};
class B: public A { public:
    void f(int) { }
    virtual int h(B *) { }
};
class C: public B { public:
    void g(double) { }
    int h(B *) { }
};
```

// Application Codes

```
A a;
B b;
C c;

A *pA;
B *pB;
```

Invocation	Initialization		
	pA = &a;	pA = &b;	pA = &c;
pA->f(2);	A::f	B::f	B::f
pA->g(3.2);	A::g	A::g	C::g
pA->h(&a);	A::h	A::h	A::h
pA->h(&b);	A::h	A::h	A::h



Binding: Exercise

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing

C Solution

Summary

```
// Class Definitions
class A { public:
    virtual void f(int) { }
    virtual void g(double) { }
    int h(A *) { }
};
class B: public A { public:
    void f(int) { }
    virtual int h(B *) { }
};
class C: public B { public:
    void g(double) { }
    int h(B *) { }
};
```

```
// Application Codes
```

```
A a;
B b;
C c;

A *pA;
B *pB;
```

Initialization			
Invocation	pB = &a;	pB = &b;	pB = &c;
pB->f(2);	Error	B::f	B::f
pB->g(3.2);	Downcast	A::g	C::g
pB->h(&a);	(A *) to	No conversion (A *) to (B *)	
pB->h(&b);	(B *)	B::h	C::h



Staff Salary Processing: Problem Statement

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing

C Solution

Summary

- An organization needs to develop a salary processing application for its staff
- At present it has an engineering division only where **Engineers** and **Managers** work. Every **Engineer** reports to some **Manager**. Every **Manager** can also work like an **Engineer**
- The logic for processing salary for **Engineers** and **Managers** are different as they have different salary heads
- In future, it may add **Directors** to the team. Then every **Manager** will report to some **Director**. Every **Director** could also work like a **Manager**
- The logic for processing salary for **Directors** will also be distinct
- Further, in future it may open other divisions, like Sales division, and expand the workforce
- **Make a suitable extensible design**



C Solution: Function Switch Engineer + Manager

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing
C Solution

Summary

- How to represent **Engineers** and **Managers**?
 - Collection of structs
- How to initialize objects?
 - Initialization functions
- How to have a collection of mixed objects?
 - Array of union
- How to model variations in salary processing algorithms?
 - struct-specific functions
- How to invoke the correct algorithm for a correct employee type?
 - Function Switch
 - Function Pointers



C Solution: Function Switch Engineer + Manager

Module 29

Partha Pratim Das

Objectives & Outline

Binding: Exercise

Staff Salary Processing

C Solution

Summary

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

typedef enum E_TYPE { Er, Mgr } E_TYPE;

typedef struct Engineer { char *name_; } Engineer;
Engineer *InitEngineer(const char *name) { Engineer *e = (Engineer *)malloc(sizeof(Engineer));
    e->name_ = strdup(name); return e;
}

void ProcessSalaryEngineer(Engineer *e) {
    printf("%s: Process Salary for Engineer\n", e->name_);
}

typedef struct Manager { char *name_; Engineer *reports_[10]; } Manager;
Manager *InitManager(const char *name) { Manager *m = (Manager *)malloc(sizeof(Manager));
    m->name_ = strdup(name); return m;
}

void ProcessSalaryManager(Manager *m) {
    printf("%s: Process Salary for Manager\n", m->name_);
}

typedef struct Staff { E_TYPE type_;
    union { Engineer *pE; Manager *pM; };
} Staff;
```




C Solution: Function Switch Engineer + Manager

Module 29

Partha Pratim Das

Objectives & Outline

Binding: Exercise

Staff Salary Processing

C Solution

Summary

```
int main() {
    Staff allStaff[10];
    allStaff[0].type_ = Er;
    allStaff[0].pE = InitEngineer("Rohit");
    allStaff[1].type_ = Mgr;
    allStaff[1].pM = InitManager("Kamala");
    allStaff[2].type_ = Mgr;
    allStaff[2].pM = InitManager("Rajib");
    allStaff[3].type_ = Er;
    allStaff[3].pE = InitEngineer("Kavita");
    allStaff[4].type_ = Er;
    allStaff[4].pE = InitEngineer("Shambhu");

    for (int i = 0; i < 5; ++i) {
        E_TYPE t = allStaff[i].type_;
        if (t == Er) ProcessSalaryEngineer(allStaff[i].pE);
        else if (t == Mgr) ProcessSalaryManager(allStaff[i].pM);
        else printf("Invalid Staff Type\n");
    }
    return 0;
}
-----
```

Output:

```
Rohit: Process Salary for Engineer
Kamala: Process Salary for Manager
Rajib: Process Salary for Manager
Kavita: Process Salary for Engineer
Shambhu: Process Salary for Engineer
```



C Solution: Function Switch Engineer + Manager + Director

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing
C Solution

Summary

- How to represent **Engineers**, **Managers**, and **Directors**?
 - Collection of structs
- How to initialize objects?
 - Initialization functions
- How to have a collection of mixed objects?
 - Array of union
- How to model variations in salary processing algorithms?
 - struct-specific functions
- How to invoke the correct algorithm for a correct employee type?
 - Function switch
 - Function pointers



C Solution: Function Switch Engineer + Manager + Director

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing
C Solution

Summary

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

typedef enum E_TYPE { Er, Mgr, Dir } E_TYPE;

typedef struct Engineer { char *name_; } Engineer;
Engineer *InitEngineer(const char *name) { Engineer *e = (Engineer *)malloc(sizeof(Engineer));
    e->name_ = strdup(name); return e;
}
void ProcessSalaryEngineer(Engineer *e) {
    printf("%s: Process Salary for Engineer\n", e->name_);
}
typedef struct Manager { char *name_; Engineer *reports_[10]; } Manager;
Manager *InitManager(const char *name) { Manager *m = (Manager *)malloc(sizeof(Manager));
    m->name_ = strdup(name); return m;
}
void ProcessSalaryManager(Manager *m) {
    printf("%s: Process Salary for Manager\n", m->name_);
}
typedef struct Director { char *name_; Manager *reports_[10]; } Director;
Director *InitDirector(const char *name) { Director *d = (Director *)malloc(sizeof(Director));
    d->name_ = strdup(name); return d;
}
void ProcessSalaryDirector(Director *d) {
    printf("%s: Process Salary for Director\n", d->name_);
}
typedef struct Staff { E_TYPE type_;
    union { Engineer *pE; Manager *pM; Director *pD; };
} Staff;
```



C Solution: Function Switch Engineer + Manager + Director

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing

C Solution

Summary

```
int main() { Staff allStaff[10];
    allStaff[0].type_ = Er;
    allStaff[0].pE = InitEngineer("Rohit");
    allStaff[1].type_ = Mgr;
    allStaff[1].pM = InitManager("Kamala");
    allStaff[2].type_ = Mgr;
    allStaff[2].pM = InitManager("Rajib");
    allStaff[3].type_ = Er;
    allStaff[3].pE = InitEngineer("Kavita");
    allStaff[4].type_ = Er;
    allStaff[4].pE = InitEngineer("Shambhu");
    allStaff[5].type_ = Dir;
    allStaff[5].pD = InitDirector("Ranjana");

    for (int i = 0; i < 6; ++i) {
        E_TYPE t = allStaff[i].type_;
        if (t == Er) ProcessSalaryEngineer(allStaff[i].pE);
        else if (t == Mgr) ProcessSalaryManager(allStaff[i].pM);
        else if (t == Dir) ProcessSalaryDirector(allStaff[i].pD);
        else printf("Invalid Staff Type\n");
    }
    return 0;
}
```

Output:

```
Rohit: Process Salary for Engineer
Kamala: Process Salary for Manager
Rajib: Process Salary for Manager
Kavita: Process Salary for Engineer
Shambhu: Process Salary for Engineer
Ranjana: Process Salary for Director
NPTEL MOOCs Programming in C++
```



C Solution: Advantages and Disadvantages

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing
C Solution

Summary

Advantages:

- Solution exists!
- Code is well structured – has patterns

Disadvantages:

- Employee data has scope for better organization
 - No encapsulation for data
 - Duplication of fields across types of employees – possible to mix up types for them (say, `char *` and `string`)
 - Employee objects are created and initialized dynamically through `Init...` functions. How to release the memory?
- Types of objects are managed explicitly by `E_Type`:
 - Difficult to extend the design – addition of a new type needs to:
 - Add new type code to `enum E_Type`
 - Add a new pointer field in `struct Staff` for the new type
 - Add a new case (`it-else`) based on the new type
 - Error prone – developer has to decide to call the right processing function for every type (`ProcessSalaryManager` for `Mgr` etc.)

Recommendation:

- Use classes for encapsulation on a hierarchy
- NPTEL MOOCs Programming in C++

Partha Pratim Das

13



Module Summary

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing
C Solution

Summary

- Practiced exercise with binding – various mixed cases
- Started designing for a staff salary problem and worked out C solutions



Instructor and TAs

Module 29

Partha Pratim
Das

Objectives &
Outline

Binding:
Exercise

Staff Salary
Processing
C Solution

Summary

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655