Bài giảng trên lớp

Bài giảng trên lớp là tài liệu gợi ý và hỗ trợ giảng viên trong quá trình lên lớp, tuy nhiên giảng viên cần có sự chuẩn bị của riêng mình cho phù hợp với từng lớp học.

Nếu giảng viên thiết kế bài giảng tốt hơn tài liệu đã cung cấp, xin hãy chủ động làm và gửi lại cho chúng tôi.

Nếu giảng viên cần thay đổi tài liệu đã cung cấp, những thay đổi có liên quan đến cấu trúc, nội dung kiến thức và có tính đổi mới, xin hãy chủ động làm và gửi lại cho chúng tôi.

Mọi ý kiến xin gửi cho Phòng NC-PTCT để được xem xét và ban hành chính thức và góp phần ngày càng hoàn thiện hơn học liệu của trường.

Trân trọng cảm ơn.

Liên hệ: Phòng NC-PTCT – FPT Polytechnic

Những ý tưởng đổi mới sẽ được xem xét và gửi qua Dự án CNGD để có sự hỗ trợ giảng viên làm nghiên cứu khoa học, hoặc hướng dẫn viết bài báo đăng trên tạp chí CNGD.

Lập trình C++
Bài 6 Lớp

Hệ thống bài cũ

- Con trỏ và tham chiếu
- Con trỏ hằng
- Con trỏ hàm
- Mảng và con trỏ

MỤC TIÊU

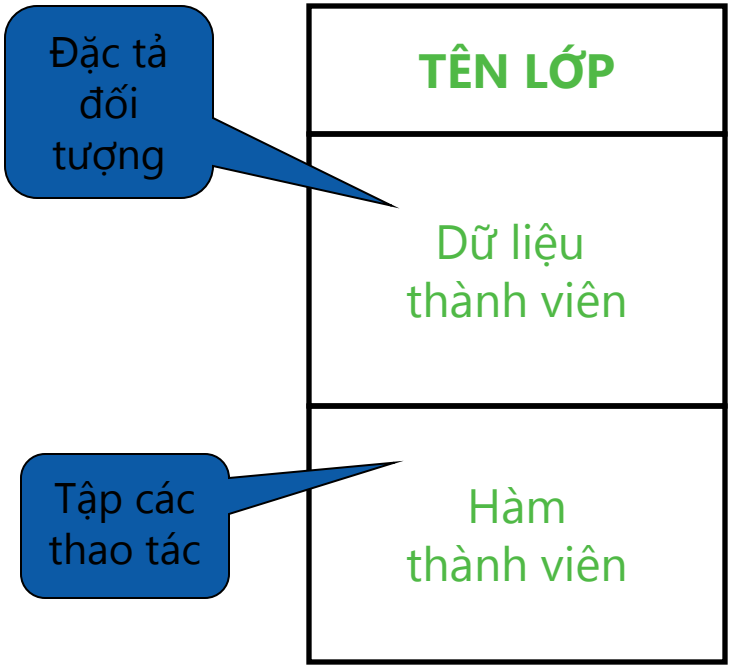
- Hiểu và cách sử dụng lớp

Nội dung

- Tổng quan về Lớp
- Hàm thành viên nội tuyến (inline)
- Hàm xây dựng, hàm hủy, hàm bạn, lớp bạn
- Đối số mặc định,toán tử phạm vi
- Danh sách khởi tạo thành viên
- Thành viên hằng, tĩnh tham chiếu
- Thành viên là đối tượng của 1 lớp
- Mảng các đối tượng
- Cấu trúc (structure) và hợp (union)

Khái niệm lớp

Lớp: kiểu dữ liệu trừu tượng.

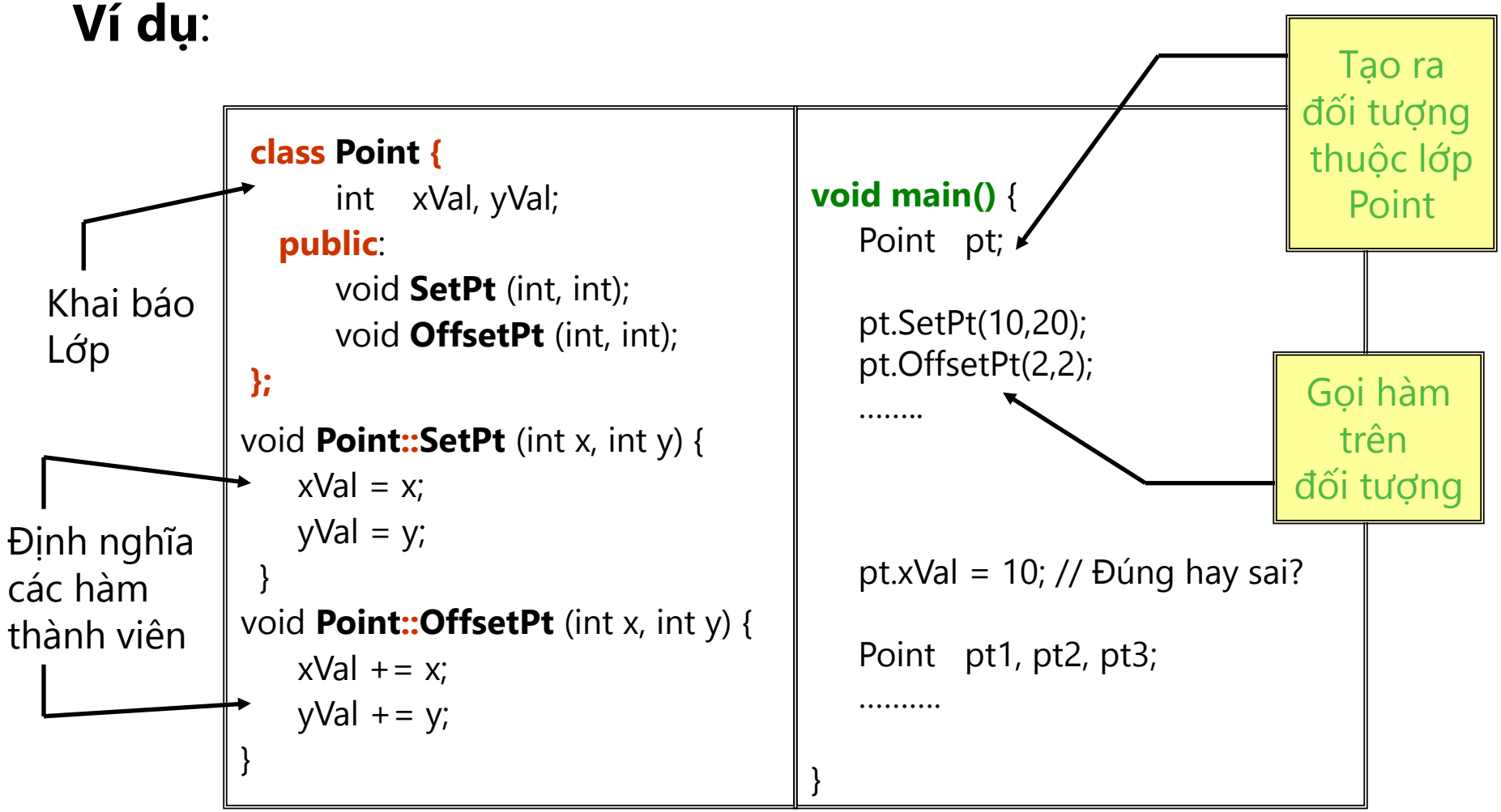


private
protected
public

```
class TÊNLỚP
   [: <Quyền truy xuất> LỚPCHA ]
 {   <Quyền truy xuất > :
          DataType1  memberdata1;
          DataType2  memberdata2;
          ..............
     < Quyền truy xuất > :
          memberFunction1();
          memberFunction2();
          ..............
 };
```

Lớp đơn giản

Ví dụ:



Phạm vi lớp

Thành viên trong 1 lớp:

- Che các thực thể trùng tên trong phạm vi.

```
// .....
int fork (void);              // fork hệ thống
class Process {
            int fork (void);  // fork thành viên
            //...
};
```

fork thành viên che đi **fork** toàn cục trong phạm vi lớp Process

```
// .....
int Process::func1 (void)
{    int  x = fork();      // gọi fork cục bộ
     int  pid = ::fork();  // gọi hàm fork hệ thống
     //...
}
```

Phạm vi lớp

Lớp toàn cục: đại đa số lớp trong C++.

Lớp lồng nhau: lớp chứa đựng lớp.

Lớp cục bộ: trong 1 hàm hoặc 1 khối.

```
class Rectangle {     // Lớp lồng nhau
      public:
            Rectangle (int, int, int, int);
            //..
      private:
            class Point {
                  public:
                        Point(int  a, int  b) { ... }
                  private:
                        int x, y;
             };
            Point   topLeft, botRight;
};
Rectangle::Point   pt(1,1);  // sd ở ngoài
```

```
void Render (Image   &i)
 {
      class ColorTable {
            public:
                  ColorTable () { /* ... */ }
                  AddEntry (int r, int g, int b)
                     { /* ... */ }
               //...
        };
      ColorTable   colors;
      //...
 }
ColorTable   ct;  // SAI
```

DEMO lớp

Hàm thành viên nội tuyến

Hàm **inline**:

- Cải thiện tốc độ thực thi
- Tốn bộ nhớ (dành cho mã lệnh) khi thực thi.

Cách 1: thêm Từ khóa **inline**

```
class Point {
        int    xVal, yVal;
    public:
        void SetPt (int, int);
        void OffsetPt (int, int);
 };
inline void Point::SetPt (int x, int y) {
     xVal = x;
     yVal = y;
 }
.....
```

Cách 2: Định nghĩa bên trong lớp

```
class Point {
     int    xVal, yVal;
public:
     void SetPt (int  x, int  y) {
          xVal = x;
          yVal = y;
      }
     void OffsetPt (int  x, int  y) {
          xVal += x;
          yVal += y;
      }
};
```

Lớp Set (tập hợp)

Ví dụ:

```
#include <iostream.h>
const  maxCard = 100;
enum   Bool {false, true};
class Set  {
      private:
            int    elems[maxCard];
            int    card;
      public:
            void    EmptySet(){ card = 0; }
            Bool    IsMember (const int);
            void    AddElem (const int);
            void    RmvElem (const int);
            void    Copy (Set&);
            Bool    Equal (Set&);
            void    Intersect (Set&, Set&);
            void    Union (Set&, Set&);
            void    Print ();
};
```

```
Bool Set::IsMember (const int elem) {
    for (register i = 0; i < card; ++i)
        if (elems[i] == elem)
            return true;
    return false;
}
void Set::AddElem (const int elem) {
    if (IsMember(elem))
        return;
    if (card < maxCard)
        elems[card++] = elem;
    else
        cout << "Set overflow"<<endl;
}
void Set::RmvElem (const int elem) {
    for (register i = 0; i < card; ++i)
        if (elems[i] == elem) {
            for (; i < card-1; ++i)  // Dịch
                elems[i] = elems[i+1];
            --card;
        }
}
```

Lớp Set (tập hợp)

Ví dụ:

```
void Set::Copy (Set &set) {
    for (register i = 0; i < card; ++i)
        set.elems[i] = elems[i];
    set.card = card;
}
Bool Set::Equal (Set &set) {
    if (card != set.card)
        return false;
    for (register i = 0; i < card; ++i)
        if (!set.IsMember(elems[i]))
            return false;
    return true;
}
void Set::Print (void) {
    cout << "{";
    for (int i = 0; i < card-1; ++i)
        cout << elems[i] << ",";
    if (card > 0)
        cout << elems[card-1];
    cout << "}"<<endl;
}
```

```
.....
int main (void) {
    Set     s1, s2;
    s1.EmptySet();  s2.EmptySet();
    s1.AddElem(10); s1.AddElem(20);
    s1.AddElem(30); s1.AddElem(40);
    s2.AddElem(30); s2.AddElem(50);
    s2.AddElem(10); s2.AddElem(60);
    cout << "s1 = ";  s1.Print();
    cout << "s2 = ";  s2.Print();
    s2.RmvElem(50);
    cout << "s2 - {50} = ";
    s2.Print();
    if (s1.IsMember(20))
        cout << "20 is in s1\n";
    if (!s1.Equal(s2))
        cout << "s1 <> s2\n";
    return 0;
}
```

DEMO lớp
tập hợp

Hàm xây dựng

- Dùng để **định nghĩa** và **khởi tạo** đối tượng cùng 1 lúc.
- Có tên trùng với tên lớp, không có kiểu trả về.
- Không gọi trực tiếp, sẽ được tự động gọi khi khởi tạo đt.
- **Gán giá trị, cấp vùng nhớ** cho các *dữ liệu thành viên*.

```
class Point {
        int    xVal, yVal;
    public:
        Point (int  x, int  y) {
             xVal = x;   yVal = y;
         }
        void OffsetPt (int  x, int  y) {
             xVal += x;  yVal += y;
         }
};
```

```
void main() {
    Point   pt1(10,20);
    pt1.OffsetPt(2,2);
    ........
  // Khai báo nào là sai ?
    Point  pt2;
    Point  pt3();
    Point  pt4 = Point(5,5);
    Point  pt5 = new Point(5,5);
    ..........
}
```

Hàm xây dựng

```
class Point {
        int    xVal, yVal;
    public:
        Point ()  // Hàm xây dựng mặc nhiên
          {  xVal = 0;  yVal = 0;  }
        Point (int  x, int  y) {
              xVal = x;   yVal = y;
          }
        Point (float  len, float  angle) {
              xVal = (int) (len * cos(angle));
              yVal = (int) (len * sin(angle));
          }
        void OffsetPt (int  , int  ); ...
};
void main() {
      Point  p1;
     Point  p2(10,20);
     Point  p3(60.3, 3.14);
}
```

```
class Set  {
    private:
        int    *elems;
        int    maxCard;
        int    card;
    public:
        Set(const int size) {
              elems = new int[size];
              maxCard = size;
              card = 0;
          }
        ..............
  };
void main() {
     Set   s1(100);
     Set   s2(20);
     Set   s3(1000); ...
}
```

Mềm dẻo hơn

Không cần phải nhớ gọi hàm EmptySet() khi khởi tạo

DEMO hàm hựng

Hàm hủy

- Dọn dẹp 1 đối tượng *trước khi* nó được thu hồi.
- Cú pháp: ~TenLop() { …… }
- Không gọi trực tiếp, sẽ được tự động gọi khi hủy bỏ đt.
- **Thu hồi vùng nhớ** cho các *dữ liệu thành viên* là *con trỏ*.

```
class Set  {
   private:
        int    *elems;
        int    maxCard;
        int    card;
   public:
        Set(const int size) { …… }
       ~Set() {  delete[]   elems;  }
        ….
};
```

```
Set  TestFunct1(Set  s1) {
        Set  *s = new Set(50);
        return *s;
   }

void main() {
     Set  s1(40), s2(50);
     s2 = TestFunct1(s1);
}
```

Tổng cộng có bao nhiêu lần hàm hủy được gọi ?

DEMO hàm hủy

Hàm bạn (Friend) – Đặt vấn đề

Tập Các Số Nguyên

```
class IntSet {
        public:
                //...
        private:
                int elems[maxCard];
                int card;
};
class RealSet {
        public:
                //...
        private:
                float elems[maxCard];
                int   card;
};
```

Tập Các Số Thực

Hàm SetToReal dùng để chuyển tập số nguyên thành tập số thực

```
void IntSet::SetToReal (RealSet &set) {
        set.card = card;
        for (register i = 0; i < card; ++i)
                set.elems[i] = (float) elems[i];
}
```

Làm thế nào để thực hiện được việc truy xuất đến thành viên **Private ?**

Hàm bạn (Friend)

Cách 1: Khai báo hàm thành viên của lớp IntSet là **bạn** (**friend**) của lớp RealSet.

Giữ nguyên định nghĩa của lớp IntSet

Thêm dòng khai báo **Friend** cho hàm thành viên **SetToReal**

```
class IntSet {
        public:
                //...
        private:
                int elems[maxCard];
                int card;
};
class RealSet {
        public:
                //...
                friend void IntSet::SetToReal (RealSet&);
        private:
                float elems[maxCard];
                int   card;
};
```

Hàm bạn (Friend)

Cách 2:

- Chuyển hàm SetToReal ra ngoài (**độc lập**).
- Khai báo hàm đó là **bạn** của cả 2 lớp.

```
class IntSet {
      public:
            //...
            friend void SetToReal (IntSet &, RealSet&);
      private:
            int elems[maxCard];
            int card;
};
class RealSet {
      public:
            //...
            friend void SetToReal (IntSet &, RealSet&);
      private:
            float elems[maxCard];
            int   card;
};
```

```
void SetToReal (IntSet&    iSet,
                RealSet& rSet )
{
   rSet.card = iSet.card;
   for (int i = 0; i < iSet.card; ++i)
     rSet.elems[i] =
          (float) iSet.elems[i];
}
```

Hàm độc lập
là bạn(friend)
của cả 2 lớp.

Hàm bạn (Friend)

Hàm bạn:

- Có quyền truy xuất đến tất cả các *dữ liệu* và *hàm* thành viên (protected + private) của 1 lớp.
- Lý do:
  - Cách định nghĩa hàm chính xác.
  - Hàm cài đặt không hiệu quả.

Lớp bạn:

- Tất cả các hàm trong lớp bạn: là hàm bạn.

```
class A;
class B { // .....
      friend class A;
};
```

```
class IntSet { ..... }
class RealSet { // .....
      friend class IntSet;
};
```

Đối số mặc định

Đối số mặc định tính từ bên phải.

```
class Point {
        int   xVal, yVal;
    public:
        Point (int x = 0, int y = 0);
        //...
};

void main() {
    Point   p1;          // như là ???
    Point   p2(10);      // như là ???
    Point   p3(10,20);
    Point   p4(, 20);  // ?????
   .....
}
```

```
class Point  {
        int   xVal, yVal;
    public:
        Point (int x = 0,  int y = 0);
        Point (float x=0, float y=0);
        //...
};

void main() {
    Point   p2(1.6, 5.0);     // như là ???
    Point   p3(10,20);        // như là ???
    Point   p4;                // ?????
   .....
}
```

Tối nghĩa
Mơ hồ

DEMO hàm bạn

Danh sách khởi tạo thành viên

Tương đương việc gán giá trị dữ liệu thành viên.

```
class Point {
        int    xVal, yVal;
    public:
        Point (int  x, int  y) {
            xVal = x;
            yVal = y;
         }
        // ......................
};
```

```
Point::Point (int  x, int  y)
          : xVal(x), yVal(y)
   {  }
```

```
class Image {
        public:
            Image(const int w, const int h);
        private:
            int   width;
            int   height;
             //...
 };
Image::Image(const int w, const int h) {
            width = w;
            height = h;
            //.....................
 }
```

```
Image::Image (const int w, const int h)
  : width(w), height(h)
 {   //.............   }
```

Thành viên hằng

Hằng dữ liệu thành viên:

```
class Image {
      public:
            Image(const int w, const int h);
      private:
            const  int    width;
            const  int    height;
   //...
};
```

Khai báo bình thường như dữ liệu thành viên

Khởi tạo **SAI**

```
class Image {
            const  int  width = 256;
            const  int  height = 168;
   //...
};
```

```
Image::Image (const int w, const int h)
            : width(w), height(h)
  {  //.....  }
```

Khởi tạo **ĐÚNG** thông qua danh sách khởi tạo thành viên

Thành viên hằng

- Hằng đối tượng: không được thay đổi giá trị.
- Hàm thành viên hằng:
  - Được phép gọi trên hằng đối tượng.
  - Không được thay đổi giá trị dữ liệu thành viên.

```
class Set {
        public:
            Set(void){ card = 0; }
            Bool    Member(const int) const;
            void    AddElem(const int);
            //...
};
Bool Set::Member (const int elem) const
 {      //...
 }
```

```
void main() {
    const Set s;
    s.AddElem(10);          // SAI
    s.Member(10);           // ok

 }
```


DEMO
DEMO thành
viên hằng

Thành viên tĩnh

Dữ liệu thành viên tĩnh:

- Dùng chung 1 bản sao chép (1 vùng nhớ) chia sẻ cho tất cả đối tượng của lớp đó.
- Sử dụng: <TênLớp>::<TênDữLiệuThànhViên>
- Thường dùng để đếm số lượng đối tượng.

```
class Window {
            // danh sách liên kết tất cả Window
      static  Window  *first;
            // con trỏ tới window kế tiếp
             Window  *next;
            //...
};

Window *Window::first = &myWindow;
// ..............
```

Khai báo

Khởi tạo dữ liệu thành viên tĩnh

Thành viên tĩnh

Hàm thành viên tĩnh:

- Tương đương với hàm toàn cục.
- Gọi thông qua: <TênLớp>::<TênHàm>

```
class Window {
        // .....
        static void PaintProc () { ..... }
        // .....
  };

void main() {
        // ..............
        Window::PainProc();

}
```

Khai báo Định nghĩa hàm thành viên tĩnh

Truy xuất hàm thành viên tĩnh


DEMO
DEMO thành
viên tĩnh

Thành viên tham chiếu

Tham chiếu dữ liệu thành viên:

```
class Image {
            int   width;
            int   height;
            int   &widthRef;
   //...
};
```

Khai báo bình thường như dữ liệu thành viên

Khởi tạo **SAI**

```
class Image {
            int   width;
            int   height;
            int   &widthRef = width;
   //...
};
```

```
Image::Image (const int w, const int h)
            : widthRef(width)
{  //..... }
```

Khởi tạo **ĐÚNG** thông qua danh sách khởi tạo thành viên

Thành viên là đối tượng của 1 lớp

Dữ liệu thành viên có thể có kiểu:

- Dữ liệu (lớp) chuẩn của ngôn ngữ.
- Lớp do người dùng định nghĩa (có thể là chính lớp đó).

```
class  Point { ……. };
class  Rectangle {
        public:
                Rectangle (int left, int top, int right, int bottom);
                //...
         private:
                Point    topLeft;
                Point    botRight;
};
Rectangle::Rectangle (int left, int top, int right, int bottom)
                : topLeft(left,top), botRight(right,bottom)
{ }
```

Khởi tạo cho các dữ liệu thành viên qua danh sách khởi tạo thành viên


DEMO
DEMO thành
viên tham chiếu

Mảng các đối tượng

Sử dụng **hàm xây dựng không đối số** (hàm xây dựng mặc nhiên - default constructor).

VD: Point pentagon[5];

Sử dụng bộ khởi tạo mảng:

VD: Point triangle[3] = {Point(4,8), Point(10,20), Point(35,15)};

Ngắn gọn:

```
Set s[4] = { 10, 20, 30, 40 };
```

tương đương với:

```
Set s[4] = { Set(10), Set(20), Set(30), Set(40) };
```

Mảng các đối tượng

Sử dụng dạng con trỏ:

- Cấp vùng nhớ:

  VD: Point *pentagon = new Point[5];

- Thu hồi vùng nhớ:

  delete[] pentagon;
  delete pentagon; *// Thu hồi vùng nhớ đầu*

```
class Polygon {
        public:
                //...
        private:
                Point    *vertices;     // các đỉnh
                int      nVertices;     // số các đỉnh
};
```

Không cần biết kích thước mảng.

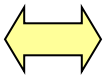
DEMO mảng
các đối tượng

Cấu trúc

Cấu trúc (structure):

- Bắt nguồn từ ngôn ngữ C.
- Tương đương với class với các thuộc tính là public.
- Sử dụng như class.

```
struct Point {
        Point (int, int);
        void OffsetPt(int, int);
        int  x, y;
};
```



```
class Point {
    public:
        Point(int, int);
        void OffsetPt(int, int);
        int          x, y;
};
```

```
Point   p = { 10, 20 };
```

Có thể khởi tạo dạng này nếu không có định nghĩa hàm xây dựng

Hợp

Hợp (union):

- Tất cả thành viên ánh xạ đến cùng 1 địa chỉ bên trong đối tượng chính nó (không liên tiếp).
- Kích thước = kích thước của dữ liệu lớn nhất.

```
union Value {
        long    integer;
        double  real;
        char    *string;
        Pair    list;
        //...
};
```

```
class Pair {
        Value   *head;
        Value   *tail;
        //...
};
```

```
class Object {
    private:
        enum ObjType {intObj, realObj,
                      strObj, listObj};
        ObjType type; // kiểu đối tượng
        Value   val;  // giá trị của đối tượng
        //...
};
```

Kích thước của Value là 8 bytes = sizeof(double)

Kết luận

- Tổng quan về Lớp
- Hàm thành viên nội tuyến (inline)
- Hàm xây dựng, hàm hủy, hàm bạn, lớp bạn
- Đối số mặc định,toán tử phạm vi
- Danh sách khởi tạo thành viên
- Thành viên hằng, tĩnh tham chiếu
- Thành viên là đối tượng của 1 lớp
- Mảng các đối tượng
- Cấu trúc (structure) và hợp (union)

Chuẩn bị bài sau

Sinh viên đọc sách và slide trước bài học kế tiếp về toán tử nạp chồng gồm:

- Nạp chồng toán tử gán
- Nạp chồng toán tử số học
- Nạp chồng toán tử quan hệ
- Nạp chồng toán tử dòng
- Nạp chồng toán tử tăng và giảm
- Nạp chồng toán tử chỉ số dưới
- Con trỏ this

Fpt University

FPT POLYTECHNIC

THANK YOU!

www.poly.edu.vn