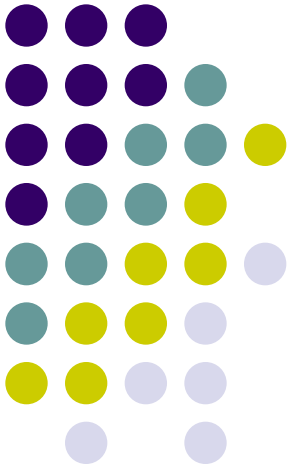
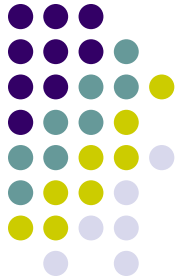
Chương 2

Những mở rộng của C++ so với C

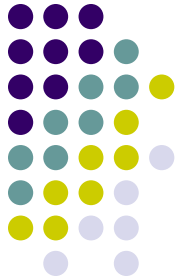


Nội dung

- Giới thiệu ngôn ngữ C++
- Mở rộng C++ so với C

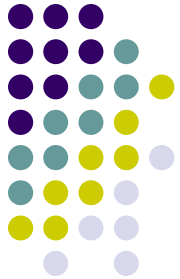


Giới thiệu ngôn ngữ C++



- Được đưa ra bởi Bjarne Stroustrup năm 1983
- Phát triển dựa trên ngôn ngữ C
- C++ có 2 đặc điểm mới:
  - Mở rộng so với C: tham chiếu, chồng hàm,…
  - Khả năng LTHĐT

Mở rộng C++ so với C

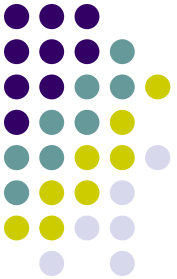


- ✓ Từ khóa mới
- ✓ Dữ liệu, khai báo biến
- ✓ Khả năng vào/ra mới
- ✓ Chuyển kiểu
- ✓ Tham chiếu
- ✓ Định nghĩa chồng hàm, tham số ngầm định
- ✓ Quản lý bộ nhớ động

Từ khóa mới

asm	**catch**	**class**
delete	**friend**	**inline**
new	**operator**	**private**
protected	**public**	**template**
this	**throw**	**try**
virtual		

Dữ liệu, khai báo biến

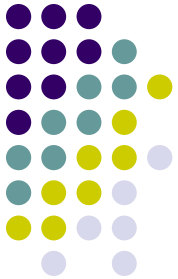


C++ cho phép khai báo biến:

- Mọi nơi
- Trước khi sử dụng

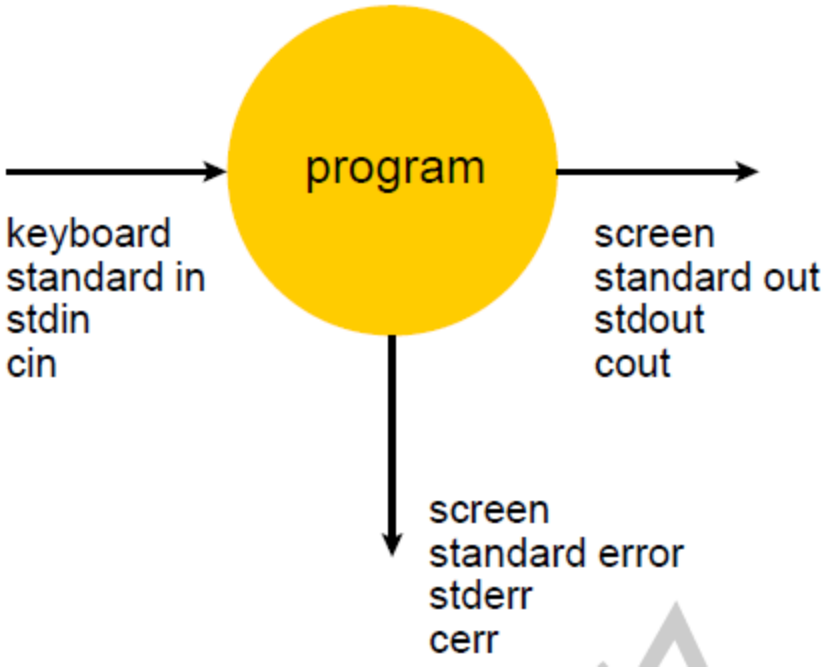
```
int n;
n = 2;
cout<<n<<“\n”;
int m = 3;
cout<<m<<“\n”;
```

Chuyển kiểu



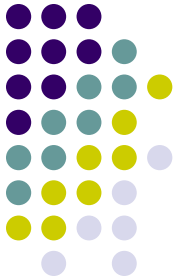
```
float m = 2.5;
int n = (int)m; //cách cũ
int n = int(m); //cách mới
```

Khả năng vào/ra mới



- C++ sử dụng stream để thực hiện thao tác vào/ra
- cin,cout, cerr là object của lớp tương ứng istream, ostream

Khả năng vào/ra mới



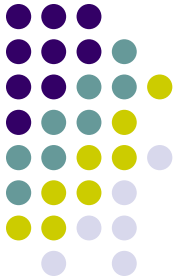
- Output

```
cout<<“Hello”;
cout<<20;
cout<<x;
```

Toán tử “<<“ có thể sử dụng nhiều lần trên 1 dòng lệnh

```
int age = 25;
cout << "Hello, I am " << age << " years old " << endl;
```

Khả năng vào/ra mới



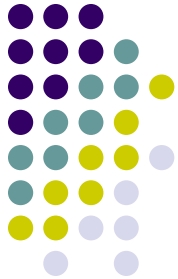
- Output

Chuyển sang dòng mới

- ✓endl
- ✓'\n'

```
int age = 25;
cout <<"First sentence.\n";
cout<<"Second sentence"<<endl;
```

Khả năng vào/ra mới



- Output

  Toán tử “<<“ có thể sử dụng để xuất ra màn hình giá trị thuộc các kiểu:

  - Hằng
  - Kiểu dữ liệu cơ bản(char, int, float, double)
  - Xâu kí tự
  - Con trỏ

Khả năng vào/ra mới

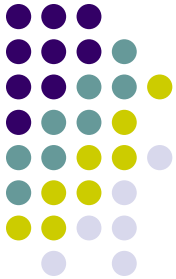
- Input

```
int age;
cin >> age;
```

Toán tử “>>“ có thể sử dụng nhiều lần trên 1 dòng lệnh

```
int  n;
float m;
char c;
cin >> n >> m >> c;
```

Khả năng vào/ra mới



- Input

  Nhập chuỗi

```
string  s;
cout << “Nhập chuỗi:”<<endl;
cin >> s;
cout << “Chuỗi đã nhập: ” << s;
```

```
Nhập chuỗi:
Hello you
Chuỗi đã nhập: Hello
```

Khả năng vào/ra mới

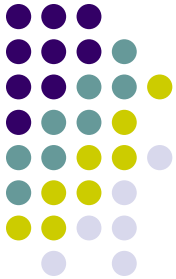
- Input

istream& getline (char* s, streamsize n);

istream& getline (char* s, streamsize n, char delim);

```
// istream getline
#include <iostream>
using namespace std;

int main () {
  char name[256], title[256];

  cout << "Enter your name: ";
  cin.getline (name,256);

  cout << "Enter your favourite movie: ";
  cin.getline (title,256);

  cout << name << "'s favourite movie is " << title;

  return 0;
}
```

Khả năng vào/ra mới



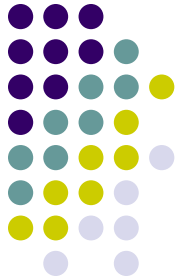
- Input

Hàm xóa bộ đệm

- fflush(stdin);
- cin.clear();

```
#include <iostream>
using namespace std;
int main()
{
    int x;
    cout << "enter choice:";
    cin >> x;
    cin.clear();
    char c;
    cin >> c;
    return 0;
}
```

Khả năng vào/ra mới

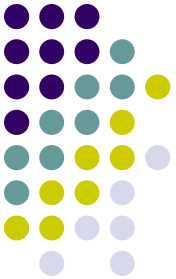


- Input

  Toán tử “>>“ để nhập dữ liệu thuộc các kiểu:

  - Kiểu dữ liệu cơ bản(char, int, float, double)
  - Xâu kí tự char*

Tham chiếu(Reference)

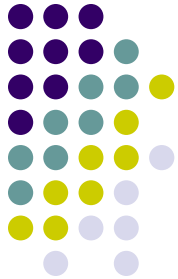


- Là “bí danh” của biến khác

```
int n = 3;
int &r = n; // r là biến tham chiếu
cout << r <<endl;
```

- Tham chiếu có thể được sử dụng là biến, tham số của hàm, giá trị trả về của hàm

Sự khác nhau giữa reference và pointer



1. Không tồn tại tham chiếu NULL

```
int  &n = NULL; //error
```

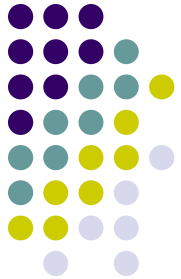
```
int *n = NULL;//OK
```

2. Tham chiếu cần phải được khởi tạo

```
string  &rs; //error
string s = “abc”;
string &rs = s;//OK
```

```
string *rs;//OK
```

Sự khác nhau giữa reference và pointer

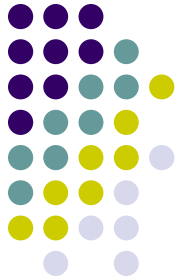


3. Tham chiếu chỉ có thể trỏ đến 1 vùng nhớ duy nhất

```
n = 6, k = 5
rn = 6, *pn = 6
n = 5, k = 5
rn = 5, *pn = 5
```

```
char s[5] = "lena";
int n = 6;
int k = 5;
cout<<"n = "<<n<<", k = "<<k<<endl;
int &rn = n;
int *pn = &n;
cout<<"rn = "<<rn<<", *pn = "<<*pn<<endl;
rn = k;
pn = &k;
cout<<"n = "<<n<<", k = "<<k<<endl;
cout<<"rn = "<<rn<<", *pn = "<<*pn<<endl;
```

Sự khác nhau giữa reference và pointer

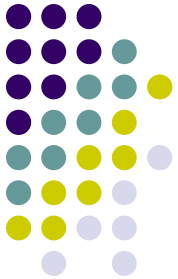


4. Tham chiếu chứa giá trị, con trỏ chứa địa chỉ của vùng nhớ trỏ đến

```
char c = 'c';
char &rn = c;
char *pn = &c;
cout<<"sizeof(rn)="<<sizeof(rn)<<", sizeof(pn)="<<sizeof(pn)<<endl;
```

```
sizeof(rn)=1, sizeof(pn)=4
```

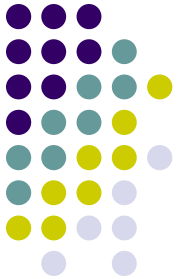
Hằng tham chiếu



Ví dụ

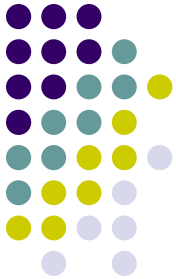
```
int  &n = 4; //error
const  int  &n = 4;

int m = 5;
const  int  &n = m;
```

Truyền tham số cho hàm bằng tham chiếu



```
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

Tham chiếu đến con trỏ



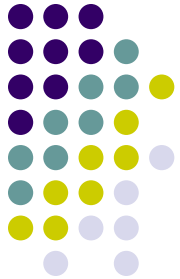
```
void function_a(int *& a) {
    *a += 5;
    int k = 7;
    a = &k;
}
```

```
void function_b(int * a) {
    *a += 5;
    int k = 7;
    a = &k;
}
```

```
int main(){
    int * myInt = new int(5);
    int * myInt2 = new int(5);
    function_a(myInt); //myInt = ?
    function_b(myInt2);  //myInt2=?
    return 0;
}
```

```
myInt = 7
myInt2 = 10
```

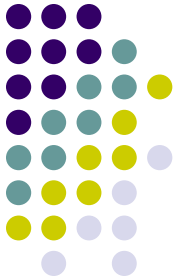
Giá trị trả về của hàm là tham chiếu



Khai báo

```
type&  func(…)
{
      ….
      return <bien co pham vi toan cuc>
}
```

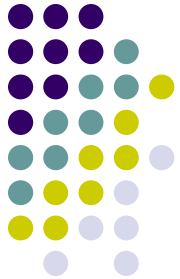
Giá trị trả về của hàm là tham chiếu



```
1  #include <iostream>
2  using namespace std;
3
4  int& get(int x)
5  {
6      int val = x*2;
7      return val;
8  }
9
10 int main()
11 {
12     int &c = get(4);
13     cout<<c<<endl;
14     return 0;
15 }
```

```
Project: sinhvien, Configuration: Debug Win32 ------

\vd_chuoi\sinhvien\main.cpp(7) : warning C4172: returning address of local variable or temporary
```

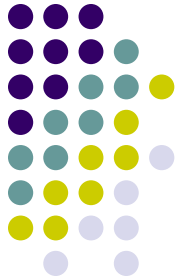
Định nghĩa chồng hàm (Overloading function)



- Sử dụng một tên cho nhiều hàm khác nhau
- Trình dịch gọi các hàm này dựa vào:
  - Số lượng tham số
  - Kiểu của tham số

Chú ý: Không dựa vào kiểu trả về của hàm

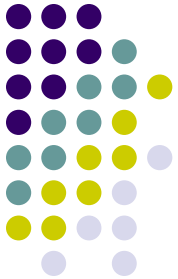
Định nghĩa chồng hàm (Overloading function)



```
2  using namespace std;
3
4  int get(int x)
5  {
6      x = x*2;
7      return x;
8  }
9  float get(int x)
10 {
11     float f = 2*x;
12     return f;
13 }
14
15 int main()
16 {
```

```
Project: sinhvien, Configuration: Debug Win32 ------

\vd_chuoi\sinhvien\main.cpp(10) : error C2556: 'float get(int)' : overloaded function differs only by return type from 'int get(int)'
```

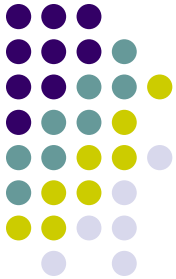
Định nghĩa chồng hàm (Overloading function)



```
1  // overloaded function
2  #include <iostream>
3  using namespace std;
4
5  int operate (int a, int b)
6  {
7    return (a*b);
8  }
9
10 float operate (float a, float b)
11 {
12   return (a/b);
13 }
14
15 int main ()
16 {
17   int x=5,y=2;
18   float n=5.0,m=2.0;
19   cout << operate (x,y);
20   cout << "\n";
21   cout << operate (n,m);
22   cout << "\n";
23   return 0;
24 }
```

```
10
2.5
```

Tham số ngầm định

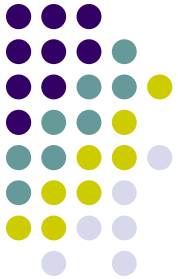


- Là tham số của hàm có giá trị mặc định
- Tham số với giá trị ngầm định phải nằm cuối cùng trong danh sách tham số

```
void print(int val1, int val2 = 20)
{
    cout << “value1 = ” << val1 <<endl;
    cout << “value2 = ” << val2 <<endl;
}
int main()
{
    print(1);
    print(3 , 4);
    return 0;
}
```

```
value1 = 1
value2 = 20
value1 = 3
value2 =  4
```

Quản lý bộ nhớ động



Toán tử cấp phát bộ nhớ new

type *pointer = new type[n]

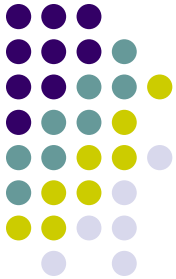
- Cấp phát vùng nhớ với kích thước n*sizeof(type)
- Ví dụ 1:

```
int *a = new int;
int *arr = new int[10];
```

- Ví dụ 2:

```
Student* student = NULL;
int n = 10;
student = new Student [n];
```

Quản lý bộ nhớ động



Toán tử giải phóng bộ nhớ delete

delete [ ] pointer;

- Toán tử delete giải phóng vùng nhớ được cấp phát bởi toán tử new
- Ví dụ 1:

```
delete a;
a = NULL;
delete [ ] arr;
arr = NULL; //tùy ý
```

- Ví dụ 2:

```
delete [ ] student; student = NULL; //tùy ý
```

Quản lý bộ nhớ động

Toán tử giải phóng bộ nhớ delete

- Nếu quên [ ] → chỉ giải phóng phần tử đầu tiên của mảng, các phần tử còn lại chưa được giải phóng nhưng không thể truy cập được
- Gán arr = NULL để đảm bảo arr không trỏ đến vùng nhớ nào