# 一、P383 超级重要

**注意：注意：注意：关于引用（红色的）返回值如果是新创的，不可以引用；如果本来就存在的全局变量，可以引用（比如改变顺序）**

**参数是引用。将参数声明为引用目的是为了提高效率。如果案值传递TIME对象，代码的功能相同，但引用传递速度更快，使用的内存少。**

**然而返回值不能用引用。因为函数将创建一个新的TIME对象（SUm），来表示另外的两个TIME对象。返回对象（如代码所做那样）将创建对象副本，而调用函数可以使用它。然而，如果返回类型为TIME&，则引用对象将是sum。但由于sum是局部变量，在函数结束后被删除，因此引用指向一个不存在的对象。使用返回类型TIME意味着程序再删除sum之前构造他的拷贝，调用函数将使用该拷贝。**

**1.1**

**const Time Time::Sum(const Time & t) const**

**{**

**Time sum;**

**sum.minutes = minutes + t.minutes;**

**sum.hours = hours + t.hours + sum.minutes / 60;**

**sum.minutes %= 60;**

**return sum;**

**}**

**1.2.**

**const Stock & Stock::topval(const Stock & s) const**

**{**

**if (s.total\_val > total\_val)**

**return s;**

**else**

**return \*this;**

**}**

// strquote.cpp -- different designs

#include <iostream>

#include <string>

using namespace std;

string version1(const string & s1, const string & s2);

const string & version2(string & s1, const string & s2); // has side effect

//const string & version3(string & s1, const string & s2); // bad design

int main()

{

string input;

string copy;

string result;

cout << "Enter a string: ";

getline(cin, input);

copy = input;

cout << "Your string as entered: " << input << endl;

result = version1(input, "\*\*\*");

cout << "Your string enhanced: " << result << endl;

cout << "Your original string: " << input << endl;

result = version2(input, "###");

cout << "Your string enhanced: " << result << endl;

cout << "Your original string: " << input << endl;

/\*

cout << "Resetting original string.\n";

input = copy;

result = version3(input, "@@@");

cout << "Your string enhanced: " << result << endl;

cout << "Your original string: " << input << endl;

// cin.get();

// cin.get();\*/

return 0;

}

string version1(const string & s1, const string & s2)

{

string temp;

temp = s2 + s1 + s2;

return temp;

}

const string & version2(string & s1, const string & s2) // has side effect

{

s1 = s2 + s1 + s2;

// safe to return reference passed to function

return s1;

}

const string & version3(string & s1, const string & s2) // bad design

{

string temp;

temp = s2 + s1 + s2;

// unsafe to return reference to local variable

return temp;

}

// swaps.cpp -- swapping with references and with pointers

#include <iostream>

void swapr(int & a, int & b); // a, b are aliases for ints

void swapp(int \* p, int \* q); // p, q are addresses of ints

void swapv(int a, int b); // a, b are new variables

int main()

{

using namespace std;

int wallet1 = 300;

int wallet2 = 350;

cout << "wallet1 = $" << wallet1;

cout << " wallet2 = $" << wallet2 << endl;

cout << "Using references to swap contents:\n";

swapr(wallet1, wallet2); // pass variables

cout << "wallet1 = $" << wallet1;

cout << " wallet2 = $" << wallet2 << endl;

cout << "Using pointers to swap contents again:\n";

swapp(&wallet1, &wallet2); // pass addresses of variables

cout << "wallet1 = $" << wallet1;

cout << " wallet2 = $" << wallet2 << endl;

cout << "Trying to use passing by value:\n";

swapv(wallet1, wallet2); // pass values of variables

cout << "wallet1 = $" << wallet1;

cout << " wallet2 = $" << wallet2 << endl;

// cin.get();

return 0;

}

void swapr(int & a, int & b) // use references

{

int temp;

temp = a; // use a, b for values of variables

a = b;

b = temp;

}

void swapp(int \* p, int \* q) // use pointers

{

int temp;

temp = \*p; // use \*p, \*q for values of variables

\*p = \*q;

\*q = temp;

}

void swapv(int a, int b) // try using values

{

int temp;

temp = a; // use a, b for values of variables

a = b;

b = temp;

}

# 二、典型记住：

//filefunc.cpp -- function with ostream & parameter

#include <iostream>

#include <fstream>

#include <cstdlib>

using namespace std;

void file\_it(ostream & os, double fo, const double fe[],int n);

const int LIMIT = 5;

int main()

{

ofstream fout;

const char \* fn = "ep-data.txt";

fout.open(fn);

if (!fout.is\_open())

{

cout << "Can't open " << fn << ". Bye.\n";

exit(EXIT\_FAILURE);

}

double objective;

cout << "Enter the focal length of your "

"telescope objective in mm: ";

cin >> objective;

double eps[LIMIT];

cout << "Enter the focal lengths, in mm, of " << LIMIT

<< " eyepieces:\n";

for (int i = 0; i < LIMIT; i++)

{

cout << "Eyepiece #" << i + 1 << ": ";

cin >> eps[i];

}

file\_it(fout, objective, eps, LIMIT);

file\_it(cout, objective, eps, LIMIT);

cout << "Done\n";

// cin.get();

// cin.get();

return 0;

}

void file\_it(ostream & os, double fo, const double fe[],int n)

{

// save initial formatting state

ios\_base::fmtflags initial;

initial = os.setf(ios\_base::fixed, ios\_base::floatfield);

std::streamsize sz = os.precision(0);

os << "Focal length of objective: " << fo << " mm\n";

os.precision(1);

os.width(12);

os << "f.l. eyepiece";

os.width(15);

os << "magnification" << endl;

for (int i = 0; i < n; i++)

{

os.width(12);

os << fe[i];

os.width(15);

os << int (fo/fe[i] + 0.5) << endl;

}

// restore initial formatting state

os.setf(initial, ios\_base::floatfield);

os.precision(sz);

}

# 三、函数重载：

// leftover.cpp -- overloading the left() function

#include <iostream>

unsigned long left(unsigned long num, unsigned ct);

char \* left(const char \* str, int n = 1);

int main()

{

using namespace std;

char \* trip = "Hawaii!!"; // test value

unsigned long n = 12345678; // test value

int i;

char \* temp;

for (i = 1; i < 10; i++)

{

cout << left(n, i) << endl;

temp = left(trip,i);

cout << temp << endl;

delete [] temp; // point to temporary storage

}

// cin.get();

return 0;

}

// This function returns the first ct digits of the number num.

unsigned long left(unsigned long num, unsigned ct)

{

unsigned digits = 1;

unsigned long n = num;

if (ct == 0 || num == 0)

return 0; // return 0 if no digits

while (n /= 10)

digits++;

if (digits > ct)

{

ct = digits - ct;

while (ct--)

num /= 10;

return num; // return left ct digits

}

else // if ct >= number of digits

return num; // return the whole number

}

// This function returns a pointer to a new string

// consisting of the first n characters in the str string.

char \* left(const char \* str, int n)

{

if(n < 0)

n = 0;

char \* p = new char[n+1];

int i;

for (i = 0; i < n && str[i]; i++)

p[i] = str[i]; // copy characters

while (i <= n)

p[i++] = '\0'; // set rest of string to '\0' （不可以省，必须吃掉余下的字符）

return p;

}

# 四、函数模板：

// twotemps.cpp -- using overloaded template functions

#include <iostream>

template <typename T> // original template

void Swap(T &a, T &b);

template <typename T> // new template

void Swap(T \*a, T \*b, int n);

void Show(int a[]);

const int Lim = 8;

int main()

{

using namespace std;

int i = 10, j = 20;

cout << "i, j = " << i << ", " << j << ".\n";

cout << "Using compiler-generated int swapper:\n";

Swap(i,j); // matches original template

cout << "Now i, j = " << i << ", " << j << ".\n";

int d1[Lim] = {0,7,0,4,1,7,7,6};

int d2[Lim] = {0,7,2,0,1,9,6,9};

cout << "Original arrays:\n";

Show(d1);

Show(d2);

Swap(d1,d2,Lim); // matches new template

cout << "Swapped arrays:\n";

Show(d1);

Show(d2);

// cin.get();

return 0;

}

template <typename T>

void Swap(T &a, T &b)

{

T temp;

temp = a;

a = b;

b = temp;

}

template <typename T>

void Swap(T a[], T b[], int n)

{

T temp;

for (int i = 0; i < n; i++)

{

temp = a[i];

a[i] = b[i];

b[i] = temp;

}

}

void Show(int a[])

{

using namespace std;

cout << a[0] << a[1] << "/";

cout << a[2] << a[3] << "/";

for (int i = 4; i < Lim; i++)

cout << a[i];

cout << endl;

}

# 五、注意：<交换>

**要const,来非const，可以；**

**要非const,来const，不可以；**

5.1（不变）

void swapv(int a, int b); // a, b are new variables

void swapv(int a, int b) // try using values

{

int temp;

temp = a; // use a, b for values of variables

a = b;

b = temp;

}

5.2

void swapr(int & a, int & b); // a, b are aliases for ints

void swapr(int & a, int & b) // use references

{

int temp;

temp = a; // use a, b for values of variables

a = b;

b = temp;

}

# 结构体引用：

//strc\_ref.cpp -- using structure references

#include <iostream>

#include <cstring>

using namespace std;

const max=30;

struct free\_throws

{

char name[max];

int made;

int attempts;

float percent;

};

void display(const free\_throws & ft);

void set\_pc(free\_throws & ft);

free\_throws & accumulate(free\_throws &target, const free\_throws &source);

int main()

{

free\_throws one = {"Ifelsa Branch", 13, 14};

free\_throws two = {"Andor Knott", 10, 16};

free\_throws three = {"Minnie Max", 7, 9};

free\_throws four = {"Whily Looper", 5, 9};

free\_throws five = {"Long Long", 6, 14};

free\_throws team = {"Throwgoods", 0, 0};

free\_throws dup;

set\_pc(one);

display(one);

accumulate(team, one);

display(team);

// use return value as argument

display(accumulate(team, two));

accumulate(accumulate(team, three), four);

display(team);

// use return value in assignment

dup = accumulate(team,five);

cout << "Displaying team:\n";

display(team);

cout << "Displaying dup after assignment:\n";

display(dup);

set\_pc(four);

// ill-advised assignment

accumulate(dup,five) = four;

cout << "Displaying dup after ill-advised assignment:\n";

display(dup);

// std::cin.get();

return 0;

}

void display(const free\_throws & ft)

{

cout << "Name: " << ft.name << '\n';

cout << " Made: " << ft.made << '\t';

cout << "Attempts: " << ft.attempts << '\t';

cout << "Percent: " << ft.percent << '\n';

}

void set\_pc(free\_throws & ft)

{

if (ft.attempts != 0)

ft.percent = 100.0f \*float(ft.made)/float(ft.attempts);

else

ft.percent = 0;

}

free\_throws & accumulate(free\_throws & target, const free\_throws & source)

{

target.attempts += source.attempts;

target.made += source.made;

set\_pc(target);

return target;

}

# 七、函数递归：

**7.1、**

// recur.cpp -- using recursion

#include <iostream>

void countdown(int n);

int main()

{

countdown(4); // call the recursive function

// std::cin.get();

return 0;

}

void countdown(int n)

{

using namespace std;

cout << "Counting down ... " << n << endl;

if (n > 0)

countdown(n-1); // function calls itself

cout << n << ": Kaboom!\n";

}

**7.2、**

// ruler.cpp -- using recursion to subdivide a ruler

#include <iostream>

const int Len = 66;

const int Divs = 6;

void subdivide(char ar[], int low, int high, int level);

int main()

{

char ruler[Len];

int i;

for (i = 1; i < Len - 2; i++)

ruler[i] = ' ';

ruler[Len - 1] = '\0';

int max = Len - 2;

int min = 0;

ruler[min] = ruler[max] = '|';

std::cout << ruler << std::endl;

for (i = 1; i <= Divs; i++)

{

subdivide(ruler,min,max, i);

std::cout << ruler << std::endl;

for (int j = 1; j < Len - 2; j++)

ruler[j] = ' '; // reset to blank ruler

}

// std::cin.get();

return 0;

}

void subdivide(char ar[], int low, int high, int level)

{

if (level == 0)

return;

int mid = (high + low) / 2;

ar[mid] = '|';

subdivide(ar, low, mid, level - 1);

subdivide(ar, mid, high, level - 1);

}

# 函数与数组array：

**8.1：array**

//arrobj.cpp -- functions with array objects

#include <iostream>

#include <array>

#include <string>

const int Seasons = 4;

const std::array<std::string, Seasons> Snames =

{"Spring", "Summer", "Fall", "Winter"};

void fill(std::array<double, Seasons> \* pa);

void show(std::array<double, Seasons> da);

int main()

{

std::array<double, 4> expenses;

fill(&expenses);

show(expenses);

// std::cin.get();

// std::cin.get();

return 0;

}

void fill(std::array<double, Seasons> \* pa)

{

for (int i = 0; i < Seasons; i++)

{

std::cout << "Enter " << Snames[i] << " expenses: ";

std::cin >> (\*pa)[i];

}

}

void show(std::array<double, Seasons> da)

{

double total = 0.0;

std::cout << "\nEXPENSES\n";

for (int i = 0; i < Seasons; i++)

{

std::cout << Snames[i] << ": $" << da[i] << '\n';

total += da[i];

}

std::cout << "Total: $" << total << '\n';

}

**8.2：一般数组：**

// arrfun3.cpp -- array functions and const

#include <iostream>

const int Max = 5;

// function prototypes

int fill\_array(double ar[], int limit);

void show\_array(const double ar[], int n); // don't change data

void revalue(double r, double ar[], int n);

int main()

{

using namespace std;

double properties[Max];

int size = fill\_array(properties, Max);

show\_array(properties, size);

if (size > 0)

{

cout << "Enter revaluation factor: ";

double factor;

while (!(cin >> factor)) // bad input

{

cin.clear();

while (cin.get() != '\n')

continue;

cout << "Bad input; Please enter a number: ";

}

revalue(factor, properties, size);

show\_array(properties, size);

}

cout << "Done.\n";

// cin.get();

// cin.get();

return 0;

}

int fill\_array(double ar[], int limit)

{

using namespace std;

double temp;

int i;

for (i = 0; i < limit; i++)

{

cout << "Enter value #" << (i + 1) << ": ";

cin >> temp;

if (!cin) // bad input

{

cin.clear();

while (cin.get() != '\n')

continue;

cout << "Bad input; input process terminated.\n";

break;

}

else if (temp < 0) // signal to terminate

break;

ar[i] = temp;

}

return i;

}

// the following function can use, but not alter,

// the array whose address is ar

void show\_array(const double ar[], int n)

{

using namespace std;

for (int i = 0; i < n; i++)

{

cout << "Property #" << (i + 1) << ": $";

cout << ar[i] << endl;

}

}

// multiplies each element of ar[] by r

void revalue(double r, double ar[], int n)

{

for (int i = 0; i < n; i++)

ar[i] \*= r;

}

# 函数与指针：

**9.1：**

// fun\_ptr.cpp -- pointers to functions

#include <iostream>

double betsy(int);

double pam(int);

// second argument is pointer to a type double function that

// takes a type int argument

void estimate(int lines, double (\*pf)(int));

int main()

{

using namespace std;

int code;

cout << "How many lines of code do you need? ";

cin >> code;

cout << "Here's Betsy's estimate:\n";

estimate(code, betsy);

cout << "Here's Pam's estimate:\n";

estimate(code, pam);

// cin.get();

// cin.get();

return 0;

}

double betsy(int lns)

{

return 0.05 \* lns;

}

double pam(int lns)

{

return 0.03 \* lns + 0.0004 \* lns \* lns;

}

void estimate(int lines, double (\*pf)(int))

{

using namespace std;

cout << lines << " lines will take ";

cout << (\*pf)(lines) << " hour(s)\n";

}

**9.2：**

// arfupt.cpp -- an array of function pointers

#include <iostream>

// various notations, same signatures

const double \* f1(const double ar[], int n);

const double \* f2(const double [], int);

const double \* f3(const double \*, int);

int main()

{

using namespace std;

double av[3] = {1112.3, 1542.6, 2227.9};

// pointer to a function

const double \*(\*p1)(const double \*, int) = f1;

// auto p2 = f2; // C++0x automatic type deduction

// pre-C++0x can use the following code instead

// const double \*(\*p2)(const double \*, int) = f2;

cout << "Using pointers to functions:\n";

cout << " Address Value\n";

cout << (\*p1)(av,3) << ": " << \*(\*p1)(av,3) << endl;

// cout << p2(av,3) << ": " << \*p2(av,3) << endl;

// pa an array of pointers

// auto doesn't work with list initialization

const double \*(\*pa[3])(const double \*, int) = {f1,f2,f3};

// but it does work for initializing to a single value

// pb a pointer to first element of pa

// auto pb = pa;

// pre-C++0x can use the following code instead

// const double \*(\*\*pb)(const double \*, int) = pa;

cout << "\nUsing an array of pointers to functions:\n";

cout << " Address Value\n";

for (int i = 0; i < 3; i++)

cout << pa[i](av,3) << ": " << \*pa[i](av,3) << endl;

cout << "\nUsing a pointer to a pointer to a function:\n";

cout << " Address Value\n";

// for (i = 0; i < 3; i++)

// cout << pb[i](av,3) << ": " << \*pb[i](av,3) << endl;

// what about a pointer to an array of function pointers

cout << "\nUsing pointers to an array of pointers:\n";

cout << " Address Value\n";

// easy way to declare pc

// auto pc = &pa;

// pre-C++0x can use the following code instead

// const double \*(\*(\*pc)[3])(const double \*, int) = &pa;

// cout << (\*pc)[0](av,3) << ": " << \*(\*pc)[0](av,3) << endl;

// hard way to declare pd

const double \*(\*(\*pd)[3])(const double \*, int) = &pa;

// store return value in pdb

const double \* pdb = (\*pd)[1](av,3);

cout << pdb << ": " << \*pdb << endl;

// alternative notation

cout << (\*(\*pd)[2])(av,3) << ": " << \*(\*(\*pd)[2])(av,3) << endl;

// cin.get();

return 0;

}

// some rather dull functions

const double \* f1(const double \* ar, int n)

{

return ar;

}

const double \* f2(const double ar[], int n)

{

return ar+1;

}

const double \* f3(const double ar[], int n)

{

return ar+2;

}

# 函数与字符串：

**10.1：**

C风格字符串

// strgback.cpp -- a function that returns a pointer to char

#include <iostream>

char \* buildstr(char c, int n); // prototype

int main()

{

using namespace std;

int times;

char ch;

cout << "Enter a character: ";

cin >> ch;

cout << "Enter an integer: ";

cin >> times;

char \*ps = buildstr(ch, times);

cout << ps << endl;

delete [] ps; // free memory

ps = buildstr('+', 20); // reuse pointer

cout << ps << "-DONE-" << ps << endl;

delete [] ps; // free memory

// cin.get();

// cin.get();

return 0;

}

// builds string made of n c characters

char \* buildstr(char c, int n)

{

char \* pstr = new char[n + 1];

pstr[n] = '\0'; // terminate string

while (n-- > 0)

pstr[n] = c; // fill rest of string

return pstr;

}

**10.2：**

String类：

// topfive.cpp -- handling an array of string objects

#include <iostream>

#include <string>

using namespace std;

const int SIZE = 5;

void display(const string sa[], int n);

int main()

{

string list[SIZE]; // an array holding 5 string object

cout << "Enter your " << SIZE << " favorite astronomical sights:\n";

for (int i = 0; i < SIZE; i++)

{

cout << i + 1 << ": ";

getline(cin,list[i]);

}

cout << "Your list:\n";

display(list, SIZE);

// cin.get();

return 0;

}

void display(const string sa[], int n)

{

for (int i = 0; i < n; i++)

cout << i + 1 << ": " << sa[i] << endl;

}