#include<iostream>

#include<string>

#include<ctime> //随机数种子

using namespace std;

实型

int main() {

//1、单精度 flout

//2、双精度 double

float f1 = 3.14f;

cout << f1 << endl;

double d1 = 3.1415926;

cout << d1 << endl;

//统计float和double的占用的内存空间

cout << "flout占用空间：" << sizeof(float) << endl;

cout << "double占用空间：" << sizeof(double) << endl;

//科学计数法

float f3 = 3e2;//3\*10^2;

cout << f3 << endl;

float f4 = 3e-2;;//3\*0.1^2;

cout << f4 << endl;

// system("pause");

return 0;

}

字符型

int main() {

//1、字符型变量创建方式

char ch = 'a';

cout << ch << endl;

ch = 97;//可以直接用ASII给字符型变量赋值

cout << ch << endl;

//1、字符型变量所占内存大小

cout << sizeof(ch) << endl;

//3、字符型变量常见的错误

// char ch2 = "b";//创建字符型变量的时候，要用单引号

char ch3 = 'acd';//创建字符型变量的时候，单引号只能有一个字母

//4、字符型变量的对应的ASCII编码

cout << (int)ch << endl;//查看字符a所对应的ASII码

//a-97

//A-65

// system("pause");

return 0;

}

//转义字符

int main() {

//1、换行符 \n

cout << "hello world\n";

//2、反斜杠 \\

cout<<"\\"<<endl;

//3、水平制表符 \t

cout << "aaa\ttt" << endl;

return 0;

}

//字符串

int main() {

//1、C风格字符串

//注意事项

//char 字符串名 []

//等号后面 要用双引号 包含起来字符串

char str[] = "hello world";

cout << str << endl;

//2\C++风格字符串 //使用前包含头文件<string>

string str2 = "hello world";

cout << str2 << endl;

return 0;

}

1

//数据的输入

int main() {

//1、整型

int a = 0;

cout << "请给整型变量a赋值："<<endl;

cin >> a;

cout <<"整型变量a = " << a << endl;

//2、浮点型

float f = 3.14f;

cout << "请给浮点型变量f赋值：" << endl;

cin>>f;

cout << "浮点型变量f = " << f << endl;

//3、字符型

char ch = 'a';

cout << "请给字符型变量f赋值：" << endl;

cin >> ch;

cout << "字符型变量f为：" << endl;

//4、字符串型

string str = "hello";

cout << "请给字符串型变量str赋值：" << endl;

cin >> str;

cout << "字符串型变量str为：" <<str<< endl;

//布尔类型

bool flag = false;

cout << "请给布尔类型flag赋值：" << endl;

cin >> flag;

cout << "布尔类型flag为：" << flag <<endl;

return 0;

}

//运算符

int main() {

//1、加减乘除

int a = 10, b = 2;

cout << a + b << endl;cout << a - b << endl; cout << a \* b << endl;cout << a / b << endl;

//两个整数相除，结果依然是整数，将小数部分去除

//2、取模运算（求余数）

int c = 3;cout << a % c << endl;

//小数不可做取模运算

//3、递增递减

//3.1 前置递增

int m = 8;++m;cout << "m = " << m << endl;

//3.2 后置递增

m++;cout << "m = " << m << endl;

//前置递增 先让变量+1，后进行表达式运算

//后置递增 先进行表达式运算，后让变量+1

int m1; m1 = m++ \* 10;//m1=80;m=9

//3.3 前置递减

--m;cout << "m = " << m << endl;

//3.4 后置递减

m--;cout << "m = " << m << endl;

//前置递减 先让变量-1，后进行表达式运算

//后置递减 先进行表达式运算，后让变量-1

//4赋值运算符

//4.1 =

int n = 10;n = 100;

cout << "n = " << n << endl;

//4.2 +=

n = 10; n += 2;

cout << "n = " << n << endl;

//4.3 -=

n = 10; n -= 2;

cout << "n = " << n << endl;

//4.4 \*=

n = 10; n \*= 2;

cout << "n = " << n << endl;

//4.4 /=

n = 10; n \*= 2;

cout << "n = " << n << endl;

//5、比较运算符

//== !=(不等于） > >= <=

//6、逻辑运算符

//6.1 非 ！ （真为假，假为真）

//6.2 与 && （同真为真，有假则假）

//6.3 或 || (有真为真，全假则假）

return 0;

}

程序流程结构：选择结构

int main() {

//1、单行 if语句

//用户输入分数，如果分数大于600，视为考上一本大学，在屏幕上输出

int score;

cout << "请输入分数：" << endl;

cin >> score; //1.1、输入分数

cout << "您输入的分数为：" << score << endl;//1.2、打印分数

if (score >= 600) {

cout << "恭喜您考上一本大学！！！" << endl;

} //1.3、判断并显示

2、多行if语句

输入考试分数，如果分数大于600，视为考上一本大学，在屏幕上输出，

如果没考上一本大学，打印未考上一本大学

int score;

cout << "请输入分数：" << endl;

cin >> score; //2.1、输入分数

cout << "您输入的分数为：" << score << endl;//2.2、打印分数

if (score >= 600) {

cout << "恭喜您考上一本大学！！！" << endl;

} //条件满足执行if；不满足执行else，

else {

cout << "未考上一本大学" << endl;

} //2.3、判断并显示

// 3、多条件if语句

//输入考试分数，如果分数大于600，视为考上一本大学，在屏幕上输出，

//如果分数小于600大于500，视为考上二本大学，在屏幕上输出

//如果分数小于500大于400，视为考上三本大学，在屏幕上输出

//如果没考上三本大学，打印未考上一本大学

int score;

cout << "请输入分数：" << endl;

cin >> score; //3.1、输入分数

cout << "您输入的分数为：" << score << endl;//3.2、打印分数

if (score >= 600) {

cout << "恭喜您考上一本大学！！！" << endl;

} //条件满足执行if；不满足执行else，

else if(score<600&&score>=500){

cout << "考上二本大学！！！" << endl;

}

else if (score < 500 && score >= 400) {

cout << "考上三本大学！！！" << endl;

}

else {

cout << "未考上大学。" << endl;

}//3.3、判断并显示

//5、嵌套if语句

//输入考试分数，如果分数大于600，视为考上一本大学，在屏幕上输出；

// //如果大于700，则考上清华大学，如果大于650，则考上北京大学；如果大于600，则考上中国人民大学

//如果分数小于600大于500，视为考上二本大学，在屏幕上输出

//如果分数小于500大于400，视为考上三本大学，在屏幕上输出

//如果没考上三本大学，打印未考上一本大学

int score;

cout << "请输入分数：" << endl;

cin >> score; //4.1、输入分数

cout << "您输入的分数为：" << score << endl;//4.2、打印分数

if (score >= 600) {

cout << "恭喜您考上一本大学！！！" << endl;

if (score > 700) {

cout << "恭喜您考上清华大学!!!" << endl;

}

else if (score > 650) {

cout << "恭喜您考上北京大学!!!" << endl;

}

else if (score > 600) {

cout << "恭喜您考上中国人民大学!!!" << endl;

}

} //条件满足执行if；不满足执行else，

else if (score < 600 && score >= 500) {

cout << "考上二本大学！！！" << endl;

}

else if (score < 500 && score >= 400) {

cout << "考上三本大学！！！" << endl;

}

else {

cout << "未考上大学。" << endl;

}//4.3、判断并显示

system("pause");

return 0;

}

// 案例：三只小猪称体重

int main() {

//创建小猪

int pig1,pig2,pig3;

//输入小猪体重

cout << "请输入小猪1的体重 ：" << endl;

cin >> pig1;

cout << "请输入小猪2的体重 ：" << endl;

cin >> pig2;

cout << "请输入小猪3的体重 ：" << endl;

cin >> pig3;

//判断

if (pig1 > pig2) {

if (pig1 > pig3) {

cout << "小猪1最重。" << endl;

}

else {

cout << "小猪3最重。" << endl;

}

}

else {

if (pig2 > pig3) {

cout << "小猪2最重。" << endl;

}

else {

cout << "小猪3最重。" << endl;

}

}

system("pause");

return 0;

}

//switch语句

int main() {

//给电影评分

//10~9 经典

//8~7 非常好

//6~5 一般

//5以下 烂片

cout << "请给电影进行打分（10~0）： \n";

int score; cin >> score;

cout << "您打的分数为 ： \n" << score << endl;;

if (score <= 10) {

switch (score) {

case 10:

cout << "您认为是经典电影!!! \n"; break;

case 9:

cout << "您认为是经典电影!!! \n"; break;

case 8:

cout << "您认为电影非常好!!! \n"; break;

case 7:

cout << "您认为电影非常好!!! \n"; break;

case 6:

cout << "您认为电影一般!!! \n"; break;

default:

cout << "您认为电影一般!!! \n"; break;

}

}

else cout << "评分错误： \n";

return 0;

}

//if和switch区别？

//switch 缺点 判断的时候只能是整型或者字符型，不可是一个区间

//swich 优点 结构清晰，执行效率高

//循环结构

//while循环语句 while（循环条件）{循环语句}

int main() {

//在屏幕中打印0~9这十个人数字

int num = 0;

while (num < 10)

//（ ）避免死循环的出现，执行循环的时候，程序必须提供跳出循环的程序

{

cout << num << endl;

num++;

}

return 0;

}

//案例：猜数字

#include<ctime>

int main() {

//系统生成随机数

//添加随机数种子，利用当前系统时间生成随机数，防止随机数都一样

//利用系统时间，需包含头文件<ctime>

srand((unsigned int)time(NULL));

int num = rand() % 100 + 1; //rand()%100+1 生成0+1~99+1的随机数

//玩家进行猜测

cout << "请进行猜数游戏："<<endl;

while (1) {

int val = 0; cin >> val;

//判断猜测

if (num > val)

{

cout << "猜小了！" << endl;

}

else if (num > val)

{

cout << "猜大了！" << endl;

}

else

{

cout << "恭喜你猜对了！！！" << endl;

break;

}

}

}

//do...while循环

int main() {

//输出0~9

int num = 0;

do {

cout << num << endl;

num++;

} while (num < 10);

//do...while和while的区别在于，前者先执行，后判断

// while (num < 10)

//

// {

// cout << num << endl;

// num++;

// }

return 0;

}

//案例：水仙花数

int main() {

//1. 打印所有三位数

int num = 100;

do {

int a = 0; int b = 0; int c = 0;

a = num % 10;// 个位

b = num / 10 % 10;//十位

c = num / 100;//百位

//2. 找到水仙花数

if (a \* a \* a + b \* b \* b + c \* c \* c == num) {

cout << num << endl;

}

num++;

}

while (num < 1000);

return 0;

}

//for循环语句

int main() {

//打印数字0~9

for (int i = 0; i < 10; i++) {//i必须重置

//1.先执行一次int i=0

//2后执行 i < 10

//3.后执行 cout << i << endl

//4.最后执行 i++ ;

//再重复2，3，4直至结束

cout << i << endl;

}

return 0;

}

//案例：敲桌子

int main() {

for (int i=0; i <= 100; i++) {

//如果是含有7或者是7的倍数，则敲桌子

if (i%10 == 7) {

cout << "敲桌子！！！" << endl;

}

else if(i % 7 == 0) {

cout << "敲桌子！！！" << endl;

}

else if (i /10 == 7) {

cout << "敲桌子！！！" << endl;

}

else {

cout << i<< endl;

}

}

return 0;

}

//嵌套循环

int main() {

//利用嵌套循环实现星图

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

cout << " \* ";

}//通常内层用j

cout << endl;

}

return 0;

}

//案例： 乘法口诀表

int main() {

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

for (int j = 1; j <=i; j++) {

cout << i << "\*" << j << "=" << i \* j << " ";

}

cout << endl;

}

return 0;

}

跳转语句

1.break使用时机（switch,循环语句 ，嵌套循环语句）

int main() {

//1.1 switch语句

cout << "请选择副本难度：" << endl;

cout << "1.普通难度！" << endl;

cout << "2.中等难度！" << endl;

cout << "3.地狱难度！" << endl;

int sel = 0;

cin >> sel;

switch (sel)

{

case 1:

cout << "您选择的是普通难度！" << endl;

break;

case 2:

cout << "您选择的是中等难度！" << endl;

break;

case 3:

cout << "您选择的是地狱难度！" << endl;

break;

default:

break;

}

//1.2 出现在for循环中

for (int i = 0; i < 10; i++) {

cout << i << endl;

if (i == 5) {

break;

}

}

//1.3 出现在嵌套循环中

for (int i = 0; i < 10; i++)

{

for (int j = 0; j <10; j++)

{

if (j==5)

{

break;

}

cout << "\* " ;

}

cout << "\* " << endl;

}

return 0;

}

//2. continue : 执行到continue就不再执行后面的代码，而是返回执行下一次的循环

int main(){

//打印0~100内的奇数

for (int i = 0; i < 100; i++){

if (i%2==0) {

continue;

}

cout << i << " ";

}

return 0;

}

//3. goto语句

int main() {

cout << "11111" << endl;

cout << "22222" << endl;

goto FLAG;

cout << "33333" << endl;

cout << "44444" << endl;

FLAG:

cout << "55555" << endl;

return 0;

}

//数组

//数组特点

// 放在一块连续的内存空间中

// 数组中的每个元素都是相同的数据类型

int main() {

//1.数据类型 数组名[数组长度]

//数组长度的下标从0开始

//int arr[5];

//arr[0] = 10;

//arr[1] = 20;

//arr[2] = 30;

//arr[3] = 40;

//arr[4] = 50;

// //访问数据元素

//cout << arr[0] << endl;

//cout << arr[1] << endl;

//cout << arr[2] << endl;

//cout << arr[3] << endl;

//cout << arr[5] << endl;

//2.数据类型 数组名[数组长度]={值1，值2，...，};

int arr2[5] = { 10,20,30,40,50 };

cout << arr2[0] << endl;

cout << arr2[1] << endl;

cout << arr2[2] << endl;

cout << arr2[3] << endl;

cout << arr2[4] << endl;

//如果在初始化数据对象，没有全部填写完成，会用0来填补剩余数值

//3.数据类型 数组名[]={值1，值2，...，};

int arr3[] = { 10,20,30,40,50,60,70,80 };

//定义数组的时候，必须有初始长度

for (int i = 0; i < 8; i++){

cout<<arr3[i]<<endl;

}

return 0;

}

//一组数组名称用途：

int main() {

//1.可以统计整个数组在内存中的长度

int arr[10] = { 1,2,3,4,5,6,7,8,9,10 };

cout << "整个数组占用内存空间为： " << sizeof(arr) << endl;

cout << "每个元素占用内存空间为： " << sizeof(arr[0]) << endl;

cout << "数组中第一个元素的地址： " << sizeof(arr) / sizeof(arr[0]) << endl;

//2.可以获取数组在内存中的首地址

cout << "数组首地址为： " << (int)arr << endl;//用int强转为十进制

cout << "数组中第一个元素的地址为：" << (int)&arr[0];

//数组名是一个常量，不可以进行复制操作

return 0;

}

//案例：五只小猪称体重

int main() {

//创建小猪体重的数组

int pig[5] = { 300,30,200,400,250 };

//找到最大值

int max = 0;

for (int i = 0; i < 4; i++) {

if (pig[i + 1] > pig[i]) {

max = pig[i + 1];

}

}

cout << "最重的小猪的体重为： " << max<< endl;

return 0;

}

//元素逆置

int main() {

//实现数组元素的逆置

//创建数组

int arr[5] = { 1,3,2,5,4 };

cout << "数组逆置前： " << endl;

for (int i = 0; i < 5; i++) {

cout << arr[i] << " ";

}

//实现逆置

int start = 0;//起始下标

int end = sizeof(arr) / sizeof(arr[0]) - 1;//结束下标

while (start<end)

{

//实现元素逆置

int temp = arr[start];

arr[start] = arr[end];

arr[end]=temp;

start++; end--;

}

cout << "数组逆置后： " << endl;

for (int i = 0; i < 5; i++) {

cout << arr[i] << " ";

}

return 0;

}

//冒泡排序

int main(){

//创建数组

int arr[] = { 4,2,8,0,5,7,1,3,9 };

cout << "排序前：";

for (int k = 0; k < 9; k++) {

cout << " " << arr[k];

}

printf("\n");

//总共排序轮数为 元素个数 - 1

for (int i = 0; i < 9-1; i++) {

//内层循环对比 次数 = 元素个数 - 当前轮数-1

for (int j = 0; j < 9-i-1; j++) {

if (arr[j+1] < arr[j]) {

int temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

cout << "排序后：" ;

for (int k = 0; k < 9; k++) {

cout << " " << arr[k];

}

return 0;

}

//二维数组

int main() {

//二维数组的定义方式

/\*

1.数据类型 数组名[ 行数 ][ 列数 ];

2.数据类型 数组名[ 行数 ][ 列数 ] = { {数据1，数据2}，{数据3，数据4} };

3.数据类型 数组名[ 行数 ][ 列数 ] = { 数据1，数据2，数据3，数据4 };

4.数据类型 数组名[ ][ 列数 ] = { 数据1，数据2，数据3，数据4 };

\*/

//1.数据类型 数组名[行数][列数];

int arr[2][2];

arr[0][0] = 1;

arr[0][1] = 2;

arr[1][0] = 3;

arr[1][1] = 4;

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 2; j++)

{

cout << arr[i][j] << " ";

}

printf("\n");

}

//2.数据类型 数组名[行数][列数] = { {数据1，数据2}，{数据3，数据4} };

int arr2[2][2] =

{

{1,2},

{3,4}

};

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 2; j++)

{

cout << arr2[i][j] << " ";

}

printf("\n");

}

//3.数据类型 数组名[行数][列数] = { 数据1，数据2，数据3，数据4 };

int arr3[2][2] = { 1,2,3,4 };

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 2; j++)

{

cout << arr3[i][j] << " ";

}

printf("\n");

}

//4.数据类型 数组名[][列数] = { 数据1，数据2，数据3，数据4 };

int arr4[][2] = { 1,2,3,4 };

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 2; j++)

{

cout << arr4[i][j] << " ";

}

printf("\n");

}

return 0;

}

//二维数组名称用途

int main() {

//可以查看二维数组所占内存空间大小

int arr[2][2] =

{

{1,2},

{3,4}

};

cout << "二维数组所占内存空间: " << sizeof(arr) << endl;

cout << "二维数组第一行所占内存空间: " << sizeof(arr[0]) << endl;

cout << "二维数组第一个元素所占内存空间: " << sizeof(arr[0][0]) << endl;

cout << "二维数组行数为： " << sizeof(arr) / sizeof(arr[0]) << endl;

cout << "二维数组列数为： " << sizeof(arr[0]) / sizeof(arr[0][0]) << endl;

//可以查看二维数组的首地址

cout << "二维数组首地址为： " <<(int) arr<< endl;

cout << "二维数组第一行首地址为： " << (int)arr[0] << endl;

cout << "二维数组第二行首地址为： " << (int)arr[1] << endl;

cout << "二维数组第一个元素首地址为： " <<(int)&arr[0][0] << endl;

cout << "二维数组第一个元素首地址为： " <<(int)&arr[0][1] << endl;

//取址符号 &

return 0;

}

// //案例：考试成绩统计

int main(){

string name [3]= {"张三","李四","王五" };

int person[3][3] =

{

{100,100,100},

{90,50,100},

{60,70,80}

};

//打印分数

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

{

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

{

cout << person[i][j] << " ";

}

printf("\n");

}

printf("\n");

//打印总分

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

{

int sum = 0;

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

{

sum += person[i][j];

}

cout << name[i] << "的总分为： " << sum << endl;

}

return 0;

}

//函数

//函数的定义

//实现一个加法函数 ： 传入两个整型数据， 计算数据相加的结果， 并返回

//1. 返回值类型 int

//2. 函数名 add

//3. 参数列表 (int sum1 , int num2 )

//4. 函数体语句 int sum = num1+ num2;

//5. runturn 表达式 ruturn sum;

//语法：

/\*返回值类型 函数名 （ 参数列表）

\* {

\* 函数体语句

\*

\* return 表达式

\*/

//定义加法函数

//函数定义的时候，num1和num2并不是真实的数据

//它只是一个形式上的参数，简称形参

int add(int num1, int num2) {

int sum = num1 + num2;

return sum;

}

int main() {

//函数的调用

//main函数中调用add函数

int a = 10;

int b = 20;

//函数调用语法 ： 函数名称 （参数）

//a和b称为 实际参数，简称实参

int c = add(a, b);

cout << "c = " << c << endl;

return 0;

}

//值传递

//定义函数，实现两个数字进行交换函数

//如果函数不需要返回值，声明的时候可以写void

void swap(int num1, int num2) {

cout << "交换前： " << endl;

cout << "num1 = " << num1 << endl;

cout << "num2 = " << num2 << endl;

int temp = num1;

num1 = num2;

num2 = temp;

cout << "交换后： " << endl;

cout << "num1 = " << num1 << endl;

cout << "num2 = " << num2 << endl;

//return: 返回值不需要的时候， 可以不写return

}

int main() {

int a = 10, b = 20;

swap(a, b);

//当我们作值传递的时候， 函数的形参发生变化， 并不会影响实参

cout << "a = " << a << endl;

cout << "b = " << b << endl;

return 0;

}

//常见的函数形式

//1. 无参无返

void test1() {

cout << "this is test1.\n";

}

//2. 有参无返

void test2(int a) {

cout << "this is test2. \n";

}

//3. 无参有返

int test3() {

cout << "this is test3. \n";

return 1000;

}

//4. 有参有返

int test4(int a) {

cout << "this is test4. \n" << "a = " << a << endl;

return a;

}

int main() {

//1. 无参无返的调用

test1();

//2. 有参无返的调用

test2(100);

//3. 无参有返的调用

int num1 = test3();

cout << "num1 = " << num1 << endl;

//4. 有参有返的调用

int num2 = test4(10000);

cout << "num2 = " << num2 << endl;

return 0;

}

//函数的声明

//比较函数， 实现两个整型数字进行比较， 返回最大值

//提前告诉编译器函数的存在， 可以利用函数的声明

//函数的声明

//int max(int a, int b);

//定义 //定义只能用一次

int max(int a, int b) {

return a > b ? a : b;

}

int main() {

int a = 10, b = 20;

cout << "a = " << a << " " << "b = " << b << endl;

cout << "max = " << max(a, b) << endl;

return 0;

}

//函数的分文件编写

//作用 ： 让代码结构更加清晰

//步骤：

//1. 创建后缀名为.h的头文件

//2.创建后缀名为.cpp的源文件

//3.在头文件中写函数的声明

//4.在源文件中写函数的定义

//实现两个数字交换的函数

////函数的声明和定义

//void swap(int a, int b) {

// int temp = a;

// a = b;

// b = temp;

//

// cout << "交换前：" << endl;

// cout << "a = " << endl;

// cout << "a = " << endl;

//}

//#include"swap.h"

int main() {

int a = 10, b = 20;

cout << "a = " << a<<endl;

cout << "b = " << b<<endl;

swap(a, b);

return 0;

}

//指针

int main() {

//1.定义指针

int a = 10;

//定义指针的语法： 数据类型 \* 指针变量名；

int\* p;

//让指针记录变量a的地址

p = &a;

cout << "a的地址为： " << &a << endl;

cout << "指针p为： " << p << endl;

//2.指针的使用

//可以通过解引用的方式来找到指针指向的内存

//指针前面加 \* 代表解引用， 找到指针指向的内存中的数据

\*p = 1000;

cout << "a= " << a << endl;

cout << "\*p= " << \*p << endl;

return 0;

}

//指针所占内存空间

int main() {

//指针所占内存空间

//32位操作系统下指针占内存大小为4个字节

//64位操作系统下指针占内存大小为8个字节

int a = 10;

int \*p;

p = &a;

cout << " sizeof (int \*) = " << sizeof(int \*) << endl;

cout << " sizeof (float \*) = " << sizeof(float \*) << endl;

cout << " sizeof (double \*) = " << sizeof(double \*) << endl;

cout << " sizeof (char \*) = " << sizeof(char \*) << endl;

return 0;

}

//空指针和野指针

//指针变量指向的内存中编号为0的空间

//用途 ：初始化指针变量

//注意 ：空指针指向的内存是不可访问的

int main() {

//空指针

//1. 空指针用于给指变量进行初始化

int\* p1 = NULL;

//2. 空指针是不可访问的

//0~255之间的内存编号是系统占用的，不允许用户访问

// cout << \*p1 << endl;

//野指针

int\* p2 = (int \*)0x1100;

//访问野指针报错

// cout << \*p2 << endl;

//空指针和野指针不可随意访问

return 0;

}

//const修饰指针

int main() {

int a = 10;

int b = 20;

int\* p = &a;

//1. const修饰指针 —— 常量指针

const int\* p = &a;

//特点 ：指针的指向可以修改， 但是指针指向的值不可以修改

\*p = 20;//错误， 指针指向的值不可以改

p = &b;//正确，指针指向可以改

//2. const修饰常量 —— 指针常量

int \*const p = &a;

//特点 ：指针指向的值可以修改， 但是指针的指向不可以修改

\*p = 20;//正确，指针指向可以改

p = &b;//错误， 指针指向的值不可以改

//3. const即修饰指针又修饰常量

const int\* const p = &a;

//特点： 指针的指向和指针指向的值都不可修改

//\*p = 20;//错误

//p = &20;//错误

return 0;

}

//指针和数组

//利用指针访问数组中的元素

int main() {

int arr1[10] = { 1,2,3,4,5,6,7,8,9,10 };

//cout << "第一个元素为： " << arr[0] << endl;

//int\* p = arr;//arr就是数组首地址

//cout << "利用指针访问第一个元素： " << \*p << endl;

//p++;//让指针向后偏移4个字节

//cout << "利用指针访问第二个元素： " << \*p << endl;

int arr2[10] = { 1,2,3,4,5,6,7,8,9,10 };

//利用指针遍历元素

int\* p = arr1; int\* q = arr2;

for (int i = 0; i < 10; i++) {

cout << "第" << \*q << "个元素为 ： " << \*p << endl;

p++; q++;

}

return 0;

}

//指针和函数

//实现两个数字进行交换

void swap1(int a, int b) {

int temp = a;

a = b;

b = temp;

cout << "swap1 a = " << a << endl;

cout << "swap1 b = " << b << endl;

}

void swap2(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

cout << "swap2 a = " << a << endl;

cout << "swap2 b = " << b << endl;

}

int main() {

//1. 值传递

int a = 10, b = 20;

swap1(a, b);

cout << "a = " <<a<< endl;

cout << "b = " <<b << endl;

//2. 地址传递

swap2(&a, &b);

cout << "a = " << a << endl;

cout << "b = " << b << endl;

return 0;

}

//指针、数组、函数

//冒泡排序函数 参数1 数组首地址 参数2 数组长度

void bubbleSort(int\* arr, int len) {

for (int i = 0; i < 10 - 1; i++) {

for (int j = 0; j < len - i - 1; j++) {

if (arr[j] < arr[j+1]) {

int temp = arr[j];arr[j] = arr[j + 1]; arr[j + 1] = temp;

}

}

}

}

void printArr(int\* arr, int len) {

for (int i = 0; i < len; i++) {

cout << arr[i] << endl;

}

}

int main() {

//1. 创建数组

int arr[10] = { 4,3,6,9,1,2,10,8,7,5 };

//2. 创建函数，实现冒泡排序

int len = sizeof(arr) / sizeof(arr[0]);

bubbleSort(arr, len);

//3. 打印结果

printArr(arr, len);

return 0;

}

//结构体

//结构体属于用户自定义的数据类型，允许用户存储不同的数据类型

//1. 创建学生数据类型 包括： 姓名 ， 年龄 ， 分数

//自定义数据类型，一些类型集合组成一个类型

//语法 struct 类型名称 { 成员列表 }

struct Student {

//成员列表

string name;//姓名

int age;//年龄

int score;//分数

}s3; // 2.3 在创建结构体时顺便创建结构体变量

int main() {

//2. 通过学生类型创建具体学生

//2.1 struct Student s1

//在C++中创建变量时，关键字可以省略

struct Student s1; //在C++中创建变量时，关键字可以省略，但定义结构体时，不可省略 //如 ： Student s1

//给s1属性赋值，通过 . 访问结构体变量中的属性

s1.name = "张三"; s1.age = 18; s1.score = 100;

cout << "姓名 ： " << s1.name << "\n年龄 ： " << s1.age << "\n分数 ： " << s1.score << endl;

//2.2 struct Struct s2 = { ... };

struct Student s2 = { "李四",19,150 };

cout << "姓名 ： " << s2.name << "\n年龄 ： " << s2.age << "\n分数 ： " << s2.score << endl;

//2.3 在创建结构体时顺便创建结构体变量

s3.name = "王五"; s3.age = 17; s3.score = 90;

cout << "姓名 ： " << s3.name << "\n年龄 ： " << s3.age << "\n分数 ： " << s3.score << endl;

return 0;

}

//结构体数组

//将自定义的结构体放到数组中方便维护

//语法： struct 结构体名 数组名[元素个数] = { {},{},...,{} };

//1. 结构体定义

struct Student {

//成员列表

string name;//姓名

int age;//年龄

int score;//分数

};

int main() {

//2. 创建结构体数组

struct Student stuArr[3] = {

{"张三",18,100},

{"李四",19,150},

{"王五",17,90}

};

//3. 给结构体数组的元素赋值

stuArr[2].name = "赵六";

stuArr[2].age = 20;

stuArr[2].score = 130;

//4. 遍历结构体数组

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

cout << "姓名 ：" << stuArr[i].name

<< " \n年龄：" << stuArr[i].age

<< " \n分数：" << stuArr[i].score << endl;

}

return 0;

}

//结构体指针

//通过指针访问结构体中的成员

//利用操作符-> 可以通过结构体指针访问结构体属性

struct Student {

//成员列表

string name;//姓名

int age;//年龄

int score;//分数

};

int main(){

//1. 创建学生结构体变量

struct Student s = { "张三",19,150 };

//2. 通过指针指向结构体变量

Student \* p = &s;

//3. 通过指针指向结构体变量

cout << "姓名 ：" << p->name

<< " \n年龄：" << p->age

<< " \n分数：" << p->score<< endl;

}

//结构体嵌套结构体

//作用： 结构体中的成员可以是另一个结构体

//

//定义学生结构体

struct Student {

//成员列表

string name;//姓名

int age;//年龄

int score;//分数

};

//定义老师结构体

struct Teacher {

int id;//职工编号

string name;//姓名

int age;//年龄

struct Student stu; //\*\*\*\*\*所嵌套的结构体必须提前给出

};

int main() {

//创建老师

Teacher t;

t.id = 10000;

t.name = "老王";

t.age = 45;

t.stu.name = "小郭";

t.stu.age = 20;

t.stu.score = 150;

cout << "老师姓名 ：" << t.name << "\n职工编号 ：" << t.id << "\n老师年龄 ：" << t.age

<< "\n所教学生姓名 ： " << t.stu.name << "\n所教学生年龄 ： " << t.stu.score << "\n所教学生分数 ： " << endl;

return 0;

}

//结构体作函数参数

//作用 ： 将结构体作为参数向函数传递

//传递方式有两种 ：1. 值传递 2.地址传递

//定义学生结构体

struct Student {

//成员列表

string name;//姓名

int age;//年龄

int score;//分数

};

//打印学生个人信息的函数

//1. 值传递

void printStudent1(struct Student s) {

cout <<"子函数打印（值传递） ： \n" << "姓名 ： " << s.name << "\n年龄 ： " << s.age << "\n分数 ： " << s.score << endl;

}

//2. 地址传递

void printStudent2(struct Student \*p) {

cout << "子函数打印（地址传递） ： \n" << "姓名 ： " << p->name << "\n年龄 ： " << p->age << "\n分数 ： " << p->score << endl;

}

int main() {

//将学生传入一个参数中， 打印学生身上的所有信息

//创建结构体变量

Student s3;

s3.name = "王五"; s3.age = 17; s3.score = 90;

// cout <<"main函数打印 ： \n" << "姓名 ： " << s3.name << "\n年龄 ： " << s3.age << "\n分数 ： " << s3.score << endl;

printStudent1(s3);

printStudent2(&s3);

//注意：

//值传递修饰形参不改变实参

//地址传递修饰形参改变实参

return 0;

}

//结构体中const的使用场景

//定义学生结构体

struct Student {

//成员列表

string name;//姓名

int age;//年龄

int score;//分数

};

void printStudent1(const Student \*p) {

//s->age=150; //\*\*加入const之后，一旦有修改操作就会报错，可以防止我们误操作

cout << "姓名 ： " << p->name << "\n年龄 ： " << p->age << "\n分数 ： " << p->score << endl;

}

int main() {

//创建结构体变量

struct Student s = { "张三",18,100 };

//打印结构体变量信息

printStudent1(&s);

return 0;

}

//结构体案例 1

//定义学生结构体

struct Student {

//成员列表

string name;//姓名

int score;//分数

};

//定义老师结构体

struct Teacher {

string name;//姓名

struct Student stu[5];

};

//给老师和学生赋值的函数

void allocateSpace(struct Teacher teacher[],int len) {

//给老师开始赋值

string name= "ABCDE";

for (int i = 0; i < len; i++) {

teacher[i].name = "Teacher\_";

teacher[i].name += name[i];

//给老师所带的学生赋值：

for (int j = 0; j < 5; j++) {

teacher[i].stu[j].name = "Student\_";

teacher[i].stu[j].name += name[j];

int random = rand() % 61 + 40;//40~99

teacher[i].stu[j].score = random;

}

}

}

//打印所有信息

void printInfo(struct Teacher teacher[],int len) {

for (int i = 0; i < len; i++) {

cout << "老师姓名：" << teacher[i].name << endl;

for (int j = 0; j < 5 ;j++)

{

cout << "\t学生姓名：" << teacher[i].stu[j].name <<

"考试分数：" << teacher[i].stu[j].score << endl;

}

}

cout << endl;

}

int main() {

//随机数种子

srand((unsigned int)time(NULL));

//1. 创建三名老师的数组

struct Teacher teacher[3];

//2. 通过函数给三名老师信息赋值，并给老师所带的学生信息赋值

int len = sizeof(teacher) / sizeof(teacher[0]);

allocateSpace(teacher, len);

//3. 打印所有老师及所带的学生信息

printInfo(teacher, len);

return 0;

}

//结构体案例 2

//设计英雄结构体

struct Hero {

//姓名

string name;

//年龄

int age;

//性别

string sex;

};

//对数组进行排序， 按照年龄进行升序排列

void bubbleSort(struct Hero Harr[],int len) {

for (int i = 0; i < len - 1; i++) {

for (int j = 0; j < len - i - 1; j++) {

if (Harr[j].age > Harr[j + 1].age)

{

struct Hero temp = Harr[j];

Harr[j] = Harr[j + 1];

Harr[j + 1] = temp;

}

}

}

}

//对排序结果进行打印输出

void printHero(struct Hero Harr[], int len) {

for (int i = 0; i < len; i++) {

cout << "\t姓名 ：" << Harr[i].name << "\t年龄 ：" << Harr[i].age << "\t性别 ："

<< Harr[i].sex << endl;

}

}

int main() {

//创建数组存放5名英雄

struct Hero Harr[5] = {

{"刘备",23,"男"},

{"关羽",22,"男"},

{"张飞",20,"男"},

{"赵云",21,"男"},

{"貂蝉",19,"女"},

};

int len = sizeof(Harr) / sizeof(Harr[0]);

cout << "排序前 ：" << endl;

for (int i = 0; i < len; i++) {

cout << "\t姓名 ：" << Harr[i].name << "\t年龄 ：" << Harr[i].age << "\t性别 ："

<< Harr[i].sex << endl;

}

bubbleSort(Harr,len);

cout << "排序后 ：" << endl;

printHero(Harr, len);

return 0;

}