1.基本的cpp源文件组成

以第一个C++程序为例子，组成部分名称见注释，详细解释见代码后

```
#include <iostream.h>  // 引库（此为包括c++标准输入输出库）
using namespace std; // 命名空间

// main函数(主函数)
int main() {

   cout << "Hello world!"<<endl; // 基本程序语句
   return 0; // 返回值
}
```

引库：其实没有引库的说法，可以叫做将库导入，前人为C++编写了很多库，比如偏向于计算机视觉的opencv库、可以用于比特处理的bitset，都属于C++的库，其中最常用的就是标准输入输出库，当然也可以自己编写库。

命名空间：用于作为附加信息来区分不同库中相同名称的函数、类、变量等，std即为标准库中使用的命名空间。比如在标准库中存在一个print()函数，fmt库中也存在一个print()函数，那么就需要使用std::print()和fmt::print()来区分。

所以，对于`cout << "Hello world!"<<endl;`，也可写为`std::cout << "Hello world!"<<std::endl;`

main函数：main() 函数是 C++ 程序的入口函数，C++ 标准规定 main() 函数的返回值类型为 int，返回值用于表示程序的退出状态，返回 0 表示程序正常退出，返回非 0，表示出现异常。

基本程序语句：即为c++执行程序的基本程序语句，实现各种功能，例子中即为打印至控制台。

返回值：返回 0 表示程序正常退出，返回非 0，表示出现异常。

2. 基本的数据类型和变量类型

使用编程语言进行编程时，需要用到各种变量来存储各种信息。变量保留的是它所存储的值的内存位置。这意味着，当您创建一个变量时，就会在内存中保留一些空间。

1. 数据类型

一共有七种基本的内置类型：

类型	关键字
布尔型	bool
字符型	char
整型	int
浮点型	float
双浮点型	double
无类型	void
宽字符型	wchar_t

可以使用下面一个或多个类型修饰符进行修饰，即为可改变类型的字节长度，从而改变可以表示的数据范围

- signed
- unsigned
- short
- long

类型	位	范围
char	1 个字节	-128 到 127 或者 0 到 255
unsigned char	1 个字节	0 到 255
signed char	1 个字节	-128 到 127
int	4 个字节	-2147483648 到 2147483647
unsigned int	4 个字节	0 到 4294967295
signed int	4 个字节	-2147483648 到 2147483647
short int	2 个字节	-32768 到 32767
unsigned short int	2 个字节	0 到 65,535
signed short int	2 个字节	-32768 到 32767
long int	8 个字节	-9,223,372,036,854,775,808 到 9,223,372,036,854,775,807
signed long int	8 个字节	-9,223,372,036,854,775,808 到 9,223,372,036,854,775,807

类型	位	范围
unsigned long int	8 个字节	0 到 18,446,744,073,709,551,615
float	4 个字节	精度型占4个字节（32位）内存空间，+/- 3.4e +/- 38 (~7 个数字)
double	8 个字节	双精度型占8 个字节（64位）内存空间，+/- 1.7e +/- 308 (~15 个数字)
long double	16 个字节	长双精度型 16 个字节（128位）内存空间，可提供18-19位有效数字。

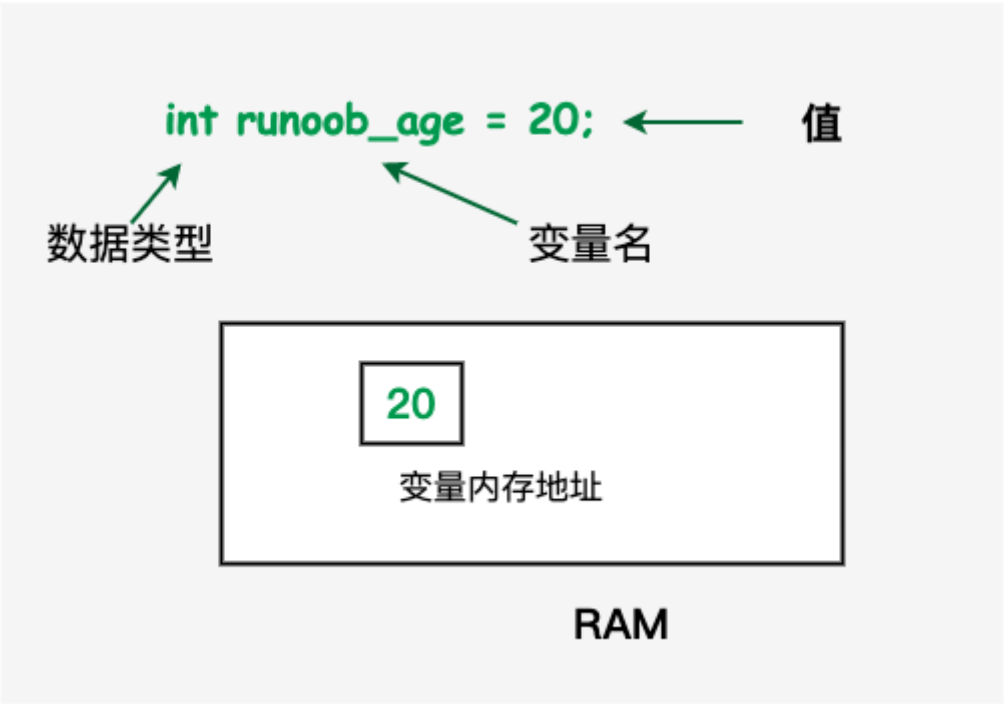
可以使用以下代码加深理解

```
#include<iostream>
#include <limits>

using namespace std;

int main()
{
    cout << "type: \t\t" << "************size**************"<< endl;
    cout << "bool: \t\t" << "所占字节数：" << sizeof(bool);
    cout << "\t最大值：" << (numeric_limits<bool>::max)();
    cout << "\t\t最小值：" << (numeric_limits<bool>::min)() << endl;
    cout << "char: \t\t" << "所占字节数：" << sizeof(char);
    cout << "\t最大值：" << (numeric_limits<char>::max)();
    cout << "\t\t最小值：" << (numeric_limits<char>::min)() << endl;
    cout << "signed char: \t" << "所占字节数：" << sizeof(signed char);
    cout << "\t最大值：" << (numeric_limits<signed char>::max)();
    cout << "\t\t最小值：" << (numeric_limits<signed char>::min)() << endl;
    cout << "unsigned char: \t" << "所占字节数：" << sizeof(unsigned char);
    cout << "\t最大值：" << (numeric_limits<unsigned char>::max)();
    cout << "\t\t最小值：" << (numeric_limits<unsigned char>::min)() << endl;
    cout << "wchar_t: \t" << "所占字节数：" << sizeof(wchar_t);
    cout << "\t最大值：" << (numeric_limits<wchar_t>::max)();
    cout << "\t\t最小值：" << (numeric_limits<wchar_t>::min)() << endl;
    cout << "short: \t\t" << "所占字节数：" << sizeof(short);
    cout << "\t最大值：" << (numeric_limits<short>::max)();
    cout << "\t\t最小值：" << (numeric_limits<short>::min)() << endl;
    cout << "int: \t\t" << "所占字节数：" << sizeof(int);
    cout << "\t最大值：" << (numeric_limits<int>::max)();
    cout << "\t最小值：" << (numeric_limits<int>::min)() << endl;
    cout << "unsigned: \t" << "所占字节数：" << sizeof(unsigned);
    cout << "\t最大值：" << (numeric_limits<unsigned>::max)();
    cout << "\t最小值：" << (numeric_limits<unsigned>::min)() << endl;
    cout << "long: \t\t" << "所占字节数：" << sizeof(long);
    cout << "\t最大值：" << (numeric_limits<long>::max)();
    cout << "\t最小值：" << (numeric_limits<long>::min)() << endl;
    cout << "unsigned long: \t" << "所占字节数：" << sizeof(unsigned long);
    cout << "\t最大值：" << (numeric_limits<unsigned long>::max)();
    cout << "\t最小值：" << (numeric_limits<unsigned long>::min)() << endl;
    cout << "double: \t" << "所占字节数：" << sizeof(double);
```

```
    cout << "\t最大值: " << (numeric_limits<double>::max)();
    cout << "\t最小值: " << (numeric_limits<double>::min)() << endl;
    cout << "long double: \t" << "所占字节数: " << sizeof(long double);
    cout << "\t最大值: " << (numeric_limits<long double>::max)();
    cout << "\t最小值: " << (numeric_limits<long double>::min)() << endl;
    cout << "float: \t\t" << "所占字节数: " << sizeof(float);
    cout << "\t最大值: " << (numeric_limits<float>::max)();
    cout << "\t最小值: " << (numeric_limits<float>::min)() << endl;
    cout << "size_t: \t" << "所占字节数: " << sizeof(size_t);
    cout << "\t最大值: " << (numeric_limits<size_t>::max)();
    cout << "\t最小值: " << (numeric_limits<size_t>::min)() << endl;
    cout << "string: \t" << "所占字节数: " << sizeof(string) << endl;
    // << "\t最大值: " << (numeric_limits<string>::max)() << "\t最小值: " <<
(numeric_limits<string>::min)() << endl;
    cout << "type: \t\t" << "************size**************"<< endl;
    return 0;
```

2. 变量类型

通过指定的数据类型，声明变量



以下为几个有效声明，作为例子：

```
int    i, j, k;
char   c, ch;
float  f, salary;
double d;
```

变量运用实例

```
#include <iostream>
using namespace std;

int main ()
{
```

```
  // 变量定义
  int a, b;
  int c;
  float f;

  // 实际初始化
  a = 10;
  b = 20;
  c = a + b;

  cout << c << endl ;

  f = 70.0/3.0;
  cout << f << endl ;

  return 0;
}

// 结果为
// 30
// 23.3333
```

3. 代码执行结构

三大类，顺序执行、循环执行、判断执行

1. 顺序执行

顾名思义，从上到下依次执行代码，不赘述

2. 循环执行

循环语句允许我们多次执行一个语句或语句组，共有四种循环类型

循环类型	描述
while循环	当给定条件为真时，重复语句或语句组。它会在执行循环主体之前测试条件。
for 循环	多次执行一个语句序列，简化管理循环变量的代码。
do...while 循环	除了它是在循环主体结尾测试条件外，其他与 while 语句类似。
嵌套循环	宁可以在 while、for 或 do..while 循环内使用一个或多个循环。

使用最多的为while和for，此处给出两个例子

```
// 使用while循环求解50-100的和
#include <iostream>
using namespace std;
int main()
{
    int sum=0,v=50;
    while (v<=100){
        sum+=v;
```

```
        ++v;
    }
    cout << "sum of 50 to 100 is "
        << sum << endl;
    return 0;
}
```

```
// 使用for循环计算1-10和
#include <iostream>
using namespace std;
int main()
{
    int sum=0;
    for (int v=1; v<=10; ++v)
        sum+=v;
    cout << "sum of 1 to 10 is "
        << sum << endl;
    return 0;
}
```

循环控制语句的使用！！

循环控制语句会更改执行的正常序列

控制语句	描述
break 语句	终止 **loop** 或 **switch** 语句，程序流将继续执行紧接着 loop 或 switch 的下一条语句。
continue 语句	引起循环跳过主体的剩余部分，立即重新开始测试条件。
goto 语句	将控制转移到被标记的语句。但是不建议在程序中使用 goto 语句。

goto为禁术，非必要不使用，不然代码会变成意大利面

3. 判断执行

语句	描述
if 语句	一个 **if 语句** 由一个布尔表达式后跟一个或多个语句组成。
if...else 语句	一个 **if 语句** 后可跟一个可选的 **else 语句**，else 语句在布尔表达式为假时执行。
if...else if...else 语句	一个 **if** 语句后可跟一个可选的 **else if...else** 语句
嵌套 if 语句	阁下可以在一个 **if** 或 **else if** 语句内使用另一个 **if** 或 **else if** 语句。
switch 语句	一个 **switch** 语句允许测试一个变量等于多个值时的情况。
嵌套 switch 语句	阁下可以在一个 **switch** 语句内使用另一个 **switch** 语句。

较为简单，直接上例子：

```
\\ if...else 语句
#include <iostream>
using namespace std;

int main ()
{
   // 局部变量声明
   int a = 100;

   // 检查布尔条件
   if( a < 20 )
   {
       // 如果条件为真，则输出下面的语句
       cout << "a 小于 20" << endl;
   }
   else
   {
       // 如果条件为假，则输出下面的语句
       cout << "a 大于 20" << endl;
   }
   cout << "a 的值是 " << a << endl;

   return 0;
}
```

```
// if...else if...else 语句
#include <iostream>
using namespace std;

int main ()
{
   // 局部变量声明
   int a = 100;

   // 检查布尔条件
   if( a == 10 )
   {
       // 如果 if 条件为真，则输出下面的语句
       cout << "a 的值是 10" << endl;
   }
   else if( a == 20 )
   {
       // 如果 else if 条件为真，则输出下面的语句
       cout << "a 的值是 20" << endl;
   }
   else if( a == 30 )
   {
       // 如果 else if 条件为真，则输出下面的语句
       cout << "a 的值是 30" << endl;
   }
   else
   {
```

```
        // 如果上面条件都不为真，则输出下面的语句
        cout << "没有匹配的值" << endl;
    }
    cout << "a 的准确值是 " << a << endl;

    return 0;
}
```

4. 作业

要求：使用visual stdio2022编写，cpp源文件命名为“作业题X”，X为题号

1. 对下列代码的结构组成进行标明（使用注释）

将代码复制到cpp源文件中会报错无法·执行，因为没有安装相关的库，注释标明结构组成即可

```
#include "main.hpp"
using namespace cv;

cv::Size dsize = Size(IMAGE_WIDTH, IMAGE_HEIGHT); // 图像大小

u_int8_t BinaryTemp[8] = {0};

cv::Mat src = cv::imread(TEST_PATH, IMREAD_COLOR); // jpeg格式
// cv::Mat src = returnRawData(); // raw格式
cv::Mat dst = Mat ::zeros(dsize, CV_64FC1);
// cv::Mat dst1 = Mat ::zeros(5, 5, CV_8UC1); //测试数组

int main()
{
  // imgPreProcessing(src, dst, dsize); // 图片预处理
  // DRAM_ApproximateStorage(src,dst, dsize);
  // DRAM_CompleteApproximateStorage(src, dst, dsize);
  // SRAM_ApproximateStorage(src, dst, dsize);
  // SRAM_CompleteApproximateStorage(src, dst, dsize);
  // PSNR_ImgApproximate(src, dst, dsize);
  // PSNR_imgCompression(src, dst, dsize);
  PSNR_CompleteImgApproximate(src, dst, dsize);
  // imwrite(RESULT_SAVE_PATH +
  //             to_string(PSNR_computing(src, dst)) +
  //             "_" +
  //             to_string(K) +
  //             "_" +
  //             IMG_NAME,
  //         dst);
  // PSNR_VddReductionAndApproximate(src, dst, dsize);
  // imwrite(RESULT_SAVE_PATH + to_string(PSNR_computing(src, dst)) + "_" +
to_string((VDD)) + ".jpeg", dst);
  // cv::imshow("rawImage", returnRawData());
  cv::imshow("src", src);
  cv::imshow("result", dst);
  cv::waitKey(0);
  return 0;
}
```

2. 给出一个循环执行中无限循环的例子，并在vs2022中运行

提交cpp源文件

3. 循环执行语句中break和continue给出相关使用例子

4. 判断执行语句中switch给出相关使用例子