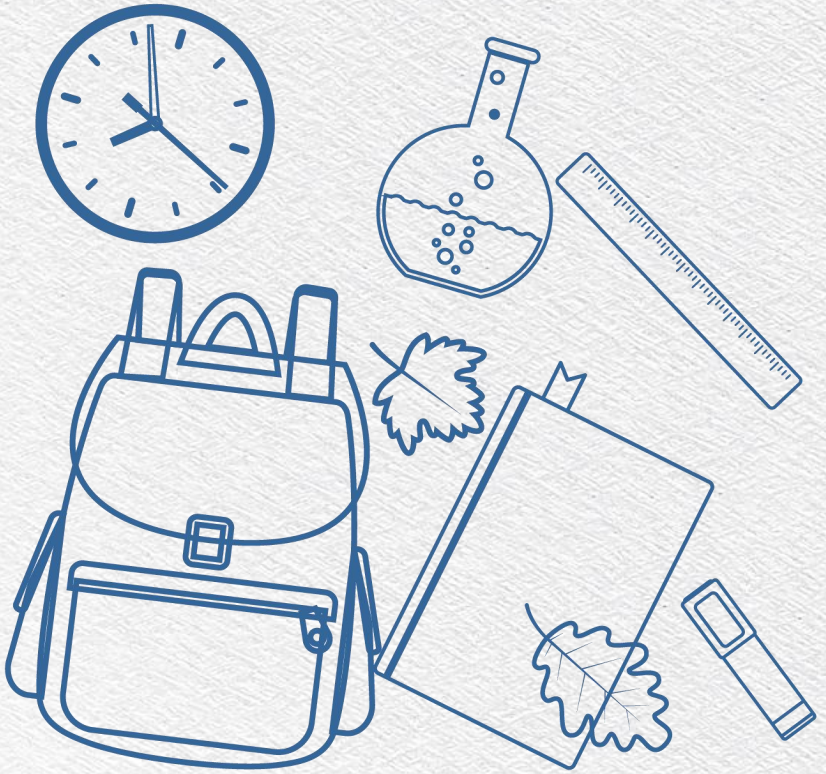
厦门市小学生信息学竞赛
初赛集训

01 语言运用与程序设计

目录

CONTENTS



Part 1

变量与数据类型

基本程序框架

```
#include<iostream>                    //头文件          #预处理符
using namspace std;                  //名空间
int main()                           //主函数
{


   return 0;                         //主函数的结束
}
```

常量与变量

常量　　const int day = 30;

变量　　int day = 30;

命名规则：

1、必须由数字、大小写字母或者下划线组成

2、数字不能开头

3、避免关键字

选一选

以下合法的变量名是（　）

A.123name

B.my name

C.MyName

D.my-name

变量的作用范围

```
#include<iostream>
using namspace std;
int a,b;      //全局变量
int main()
{
   int a;              //局部变量:在定义的局部作用域{}内使用

   return 0;
}
```

选一选

局部变量和全局变量同名时，在局部作用域内优先使用的是（ ）

A. 全局变量

B. 局部变量

C. 取决于变量的类型

D. 会产生编译错误

选一选

以下哪种变量的作用域仅限于定义它的函数内部？（ ）

A. 全局变量

B. 局部变量

C. 静态变量

D. 成员变量

选一选

```
int a = 10,b = 5;
int main()
{
   int a = 4,c;
   c = a + b;
   cout << c;
   return 0;
}
```

上面程序的运行输出结果是（ ）

A. 9　　B. 10　　C. 15　　D. 报错

基本数据类型

数据类型	标识符	内存大小	数据范围
整型	**int**	**4Byte = 32bits**	**-2147483648~2147483647**
长整型	**long long**	8Byte = 64bits	-2^63 ~ 2^63 -1
单精度浮点数	float	4 Byte	-3.4E+38~3.4E+38(7 位有效数字)
双精度浮点数	double	8 Byte	-1.79E+308~1.79E+308(15 位有效数字)
字符型	**char**	**1 Byte**	**-128~127**
布尔型	**bool**	**1 Byte**	0 或 1

选一选

```
int main()
{
   int a = 9,b = 2;
   double c = a / b;
   cout << c;
   return 0;
}
```

上面程序的运行输出结果是（ ）

A. 4　　B. 4.0　　C. 4.000000　　D. 4.5

选一选

```
int main()
{
   double a = 9,b = 2;
   int c = a / b;
   cout << c;
   return 0;
}
```

上面程序的运行输出结果是（ ）

A. 4　　B. 4.0　　C. 4.000000　　D. 4.5

Part 2

输入输出方法

C++的输入和输出(头文件<iostream>)

cin	输入
cout	输出
<< >>	流运算符
endl	换行

使用范例:

cin >> 变量 >> 数组;

cout << 数字 << 式子 << “字符串” << 变量 << endl;

C的输入和输出(头文件<cstdio>)

scanf(“格式控制字符串”,&变量);

例如:

scanf(“%d : %d”,&a,&b)

输入：10:08

实现：10 → a　08 → b

C的输入和输出(头文件<cstdio>)

printf(“格式控制字符串”,变量);

例如:

scanf(“I am %d years old”,a)

实现：a = 8

输出：I am 8 years old

常用占位符

%d	整型
%lld	长整型
%u	无符号数
%o	八进制整数
%x	十六进制整数
%f	浮点数，默认6位小数
%.2f	保留两位小数
%c	字符
%s	字符串

填一填

```
int a = 10;
float b = 3.14;
char c = 'A';
printf("a = %d, b = %f, c = %c", a, b, c);
```

上面程序的输出结果是_____

Part 3

基本运算符

算术运算符

算术运算	符号
加法	+
减法	-
乘法	*
除法	/
取余	%

关系运算符

关系运算	符号
大于	>
小于	<
等于	==
不等于	!=
大于等于	>=
小于等于	<=

逻辑运算符

逻辑运算	符号
且	&&
或	||
非	!

下面运算的结果分别是?

5 < 3 || 10 > 7 && 2 == 1

5 > (a = 3) && 5 * (3 - 2) < 5

!(10 == 10) + 2 > !5 + 1

复合运算符

a += b	==>	a = a + b
a -= b	==>	a = a - b
a *= b	==>	a = a * b
a /= b	==>	a = a / b
a %= b	==>	a = a % b

其他运算符

自增运算符：++ --

a++：先使用a的值，再使a增加1

++a：先使a增加1，再使用a的值

三目运算符： A？B：C

条件A是否成立，若成立使用B的值，若不成立使用C的值

选一选

```
#include <stdio.h>
int main() {
    int a = 5;
    int b = ++a + a++;
    printf("%d, %d", a, b);
    return 0;
}
```

上面程序的运行输出结果是（ ）

A. 6 11　　**B. 7 11**

C. 7 12　　**D. 7 13**

选一选

```
int main() {
    int i = 0, j = 0;
    while (i++ < 5) {
        j += i;
    }
    printf("%d, %d", i, j);
    return 0;
}
```

上面程序的运行输出结果是（ ）

A. 5 11　　**B. 5 15**

C. 6 15　　**D. 6 21**

填一填

```
#include <stdio.h>
int main() {
    int x = 10, y = 20;
    int z = (x > y)? x++ : y++;
    printf("x = %d, y = %d, z = %d", x, y, z);
    return 0;
}
```

上面程序的运行输出结果是_____

运算符的优先级

优先级	运算符	运算方向
1	[] () . ->	左到右
2	-(负号) +(正号) ++ -- *(指针) & ！ ~	右到左
3	* / %	左到右
4	+ -	左到右
5	== != >= <= > <	左到右
6	&&(&&高于||) ||	左到右
7	？：	右到左

下面程序的结果是_____

```
#include <stdio.h>

int main() {
    int a = 5, b = 8, c = 3;
    int result;
    result = (a < b)? (a + c) : (b - c);
    result = (result >= 8)? result++ : result--;
    printf("%s", (result == a * b)? "Equal" : "Not Equal");
    return 0;
}
```

下面程序的结果是_____

```
#include <stdio.h>

int main() {
    int x = 4, y = 6, z = 2;
    int result;
    result = (x > y && x < z)? x++ : (y > z && y < x)?
y++ : z++;
    printf("%d, %d, %d, %d", result, x, y, z);
    return 0;
}
```

下面程序的结果是_____

```
#include <stdio.h>

int main() {
    int a = 3, b = 5, c = 7;
    int result;
    result = (a < b || a > c)? ((a + b > c)? a + b : c) :
((b + c > a)? b + c : a);
    printf("result = %d", result);
    return 0;
}
```

Part 4

程序结构设计

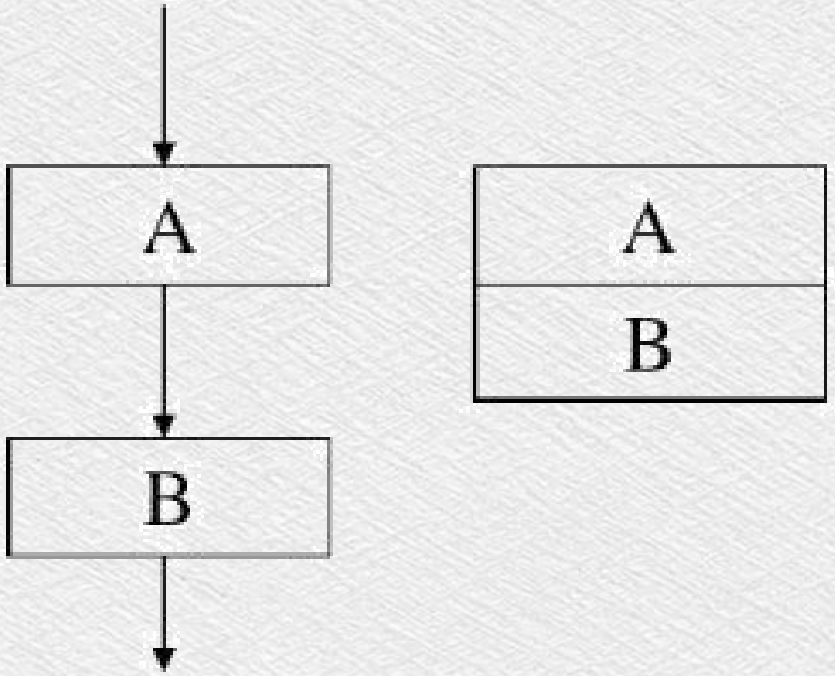
结构化程序设计

基本思想：任何程序都可以用三种基本结构表示

结构化程序：结构化程序设计采用自顶向下、逐步求精的设计方法，各个模块通过“顺序、选择、循环”的控制结构进行连接，并且只有一个入口、一个出口。

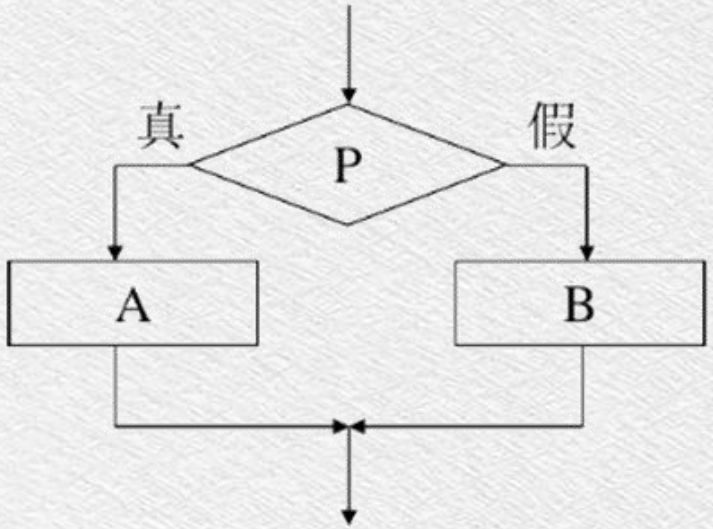
三种基本结构

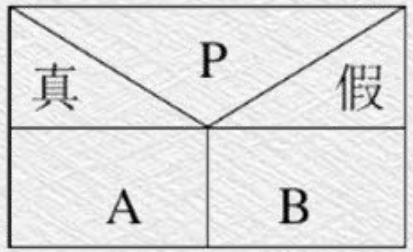
顺序结构：顺序结构表示程序中的各操作是按照它们出现的先后顺序执行的。

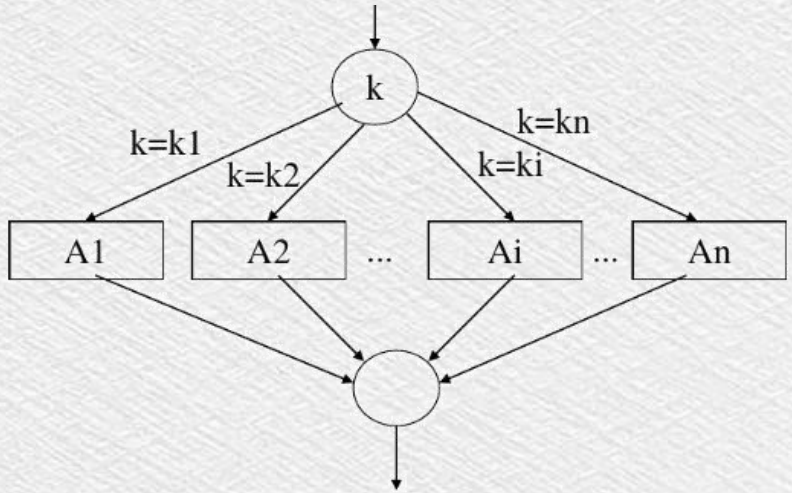


三种基本结构

选择结构：选择结构表示程序的处理步骤出现了分支，它需要根据某一特定的条件选择其中的一个分支执行。选择结构有单选择、双选择和多选择三种形式。







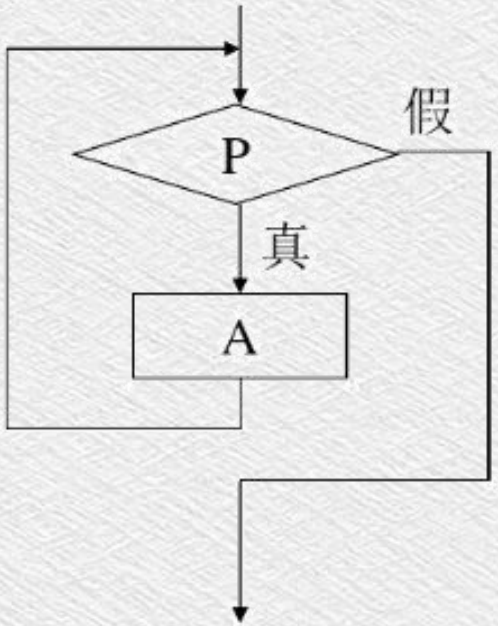
三种基本结构

循环结构：循环结构表示程序反复执行某个或某些操作，直到某条件为假（或为真）时才可终止循环。

循环结构的基本形式有两种：当型循环和直到型循环。

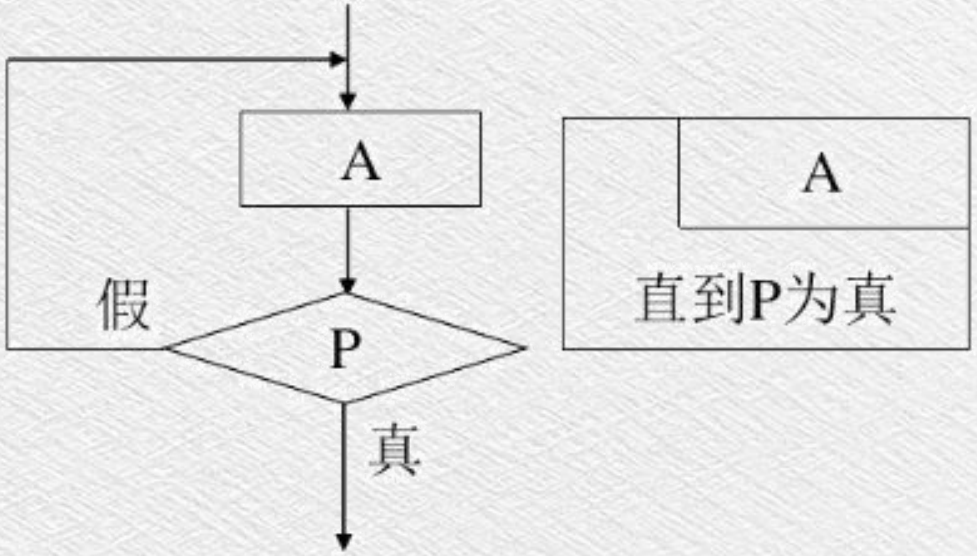
当型循环

表示先判断条件，当满足给定的条件时执行循环体，并且在循环终端处流程自动返回到循环入口；如果条件不满足，则退出循环体直接到达流程出口处。因为是"当条件满足时执行循环"，即先判断后执行，所以称为当型循环。



直到型循环

表示从结构入口处直接执行循环体，在循环终端处判断条件，如果条件不满足，返回入口处继续执行循环体，直到条件为真时再退出循环到达流程出口处，是先执行后判断。因为是"直到条件为真时为止"，所以称为直到型循环。



程序框图

程序框图属于流程图，是一种用规定的图形、流程线及文字说明来准确表示算法的图形，具有直观、形象的特点，能清楚地展现算法的逻辑结构。

画程序框图的规则

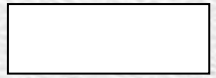
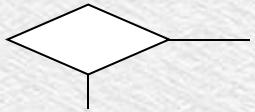
a．使用标准的框图符号。

b．一般按从上到下、从左到右的方向画。

c．除判断框外，大多数框图符号只有一个进入点和一个退出点。

d．一种判断框是二择一形式的判断，有且仅有两个可能结果；另一种是多分支判断，可能有几种不同的结果。

e．在图形符号内描述的语言要非常简练清楚。

图形符号	名称	符号表示的意义
	起、止框	框图的开始或结束
	输入、输出框	数据的输入或者结果的输出
	处理框	赋值、执行计算语句、结果的传送
	判断框	根据给定条件判断
	流程线	流程进行的方向
	连结点	连结另一页或另一部分的框图

选一选

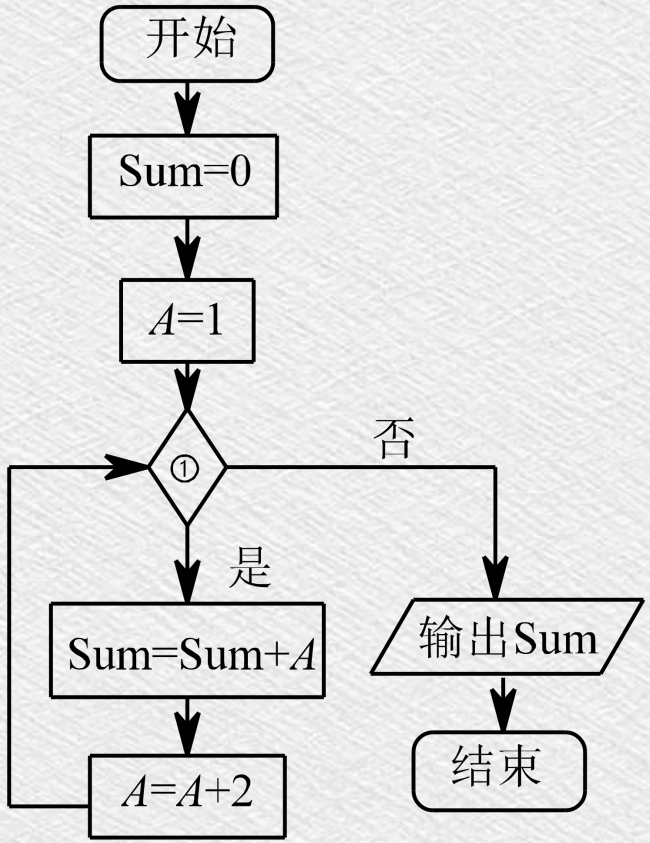
执行如图所示是求Sum=1+3+…+101的程序框图，其中①应该为（　）。

A．A = 101?

B．A ≤ 101?

C．A > 101?

D．A ≥ 101?



选一选

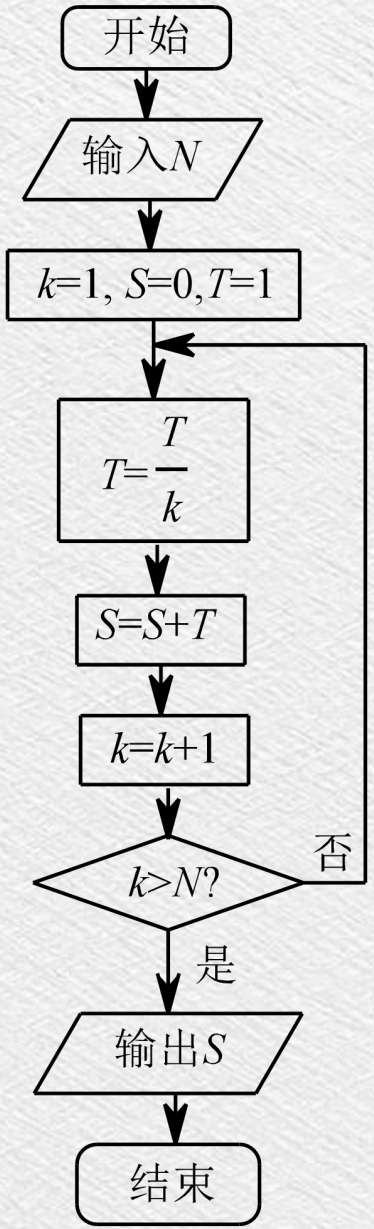
执行如图所示的程序框图，如果输入的N=4，那么输出的S等于（　）。

A. $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}$

B. $1+\frac{1}{2}+\frac{1}{3\times2}+\frac{1}{4\times3\times2}$

C. $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}$

D. $1+\frac{1}{2}+\frac{1}{3\times2}+\frac{1}{4\times3\times2}+\frac{1}{5\times4\times3\times2}$



函数

函数是一组一起执行一个任务的语句。每个C++程序都至少有一个函数，即主函数 main()，所有简单的程序都可以定义其他额外的函数。但在逻辑上，划分通常是根据每个函数执行一个特定的任务来进行的。

函数声明告诉编译器函数的名称、返回类型和参数。函数定义提供了函数的实际主体。

函数的一般形式

C++ 中的函数定义的一般形式如下：

```
返回值类型  函数名(形式参数) {

        函数体;

        return 返回值;

}
```

在C++中，函数由一个函数头和一个函数主体组成。

函数的组成部分

返回类型　　函数名　　参数

```
int add(int num1, int num2)
{
    int sum = num1 + num2;
    return sum;
}
```

函数主体

函数的组成部分

有些函数执行所需的操作而不返回值，在这种情况下，返回值类型是关键字 **void**。

函数名和参数列表一起构成了**函数签名**，只要参数列表不同，函数名允许重名。

函数参数的传递

当函数被调用时，你向参数传递一个值，这个值被称为**实际参数**，它被存在**形式参数**中。参数列表包括函数参数的类型、顺序、数量。

参数个数是可选的，也就是说，函数可能不包含参数，也可能包括多个参数。形式参数可以看做函数的局部变量。

函数参数的传递

程序中，实参向形参是单向值传递的方式，并且普通变量作为函数参数，实参将值传递给形参。

指针（包括数组名）作为函数参数，由于指针变量和数组名的值均为地址，因此实参和形参之间传递的是地址，其结果是形参指向了实参所指的地址。如：

函数参数的传递

```
void swap(int* x, int* y)  {
        int temp;
        temp = *x;
        *x = *y;
        *y = temp
}
```

主函数：

```
    int a = 5, b = 10;
    swap(a, b);
```

函数参数的传递

```
void swap(int x, int y) {
        int temp;
        temp=x;
        x=y;
        y=temp
    }
```

主函数：

```
    int a=5,b=10;
    swap(a,b);
```