一、

1. C
2. C
3. A
4. C
5. B
6. A
7. D
8. B
9. B
10. C

二、

- 6　　　　5
- 字母或下划线
- float
- 43
- 1
- a*(b*(c*(d+x*x)+y)+z)
- 3
- 1　　　　0

三、

1. i=30
2. 1#2#3#4#
3. 3456789012
4. !yppah
5. 60,2,30,2,15,3,5,5,

四、

1、

(1)i<N

(2)i<N

(3)<N-i

(4)a[j]>a[j+1]

(5)a[j]=a[j+1]

2、

(6)&n

(7)m%10

(8)m/10

(9)m>0

(10)r(m)

3、

(11)a[i]+b[i]

(12)a[i+1]

(13)c++

(14)*cp

(15)c-1

五、

1、

```c
#include <stdio.h>
int main(void) {
 int n,x;
 int T(int n, int x, FILE *file);
 printf("请输入n,x:\n");
 scanf("%d%d", &n, &x);
 FILE *file;
 if((file=fopen("tdat.txt","w"))==NULL){
 printf("打开文件失败\n");
  return 0;
  }
  for (int i = 0; i <= n; ++i) {
  fprintf(file, "T%d(%d)=%d\n", i, x, T(i, x, file));
  }

  return 0;
 }
 int T(int n, int x, FILE *file){
  int t=0;
  if (n==0){
  t=1;
  return t;
  } else if(n==1){
  t=x;
  return t;
  }
  t = 2 * x * T(n - 1, x, file) - T(n - 2, x, file);
  return t;
 }
```

2、

```
#include <stdio.h>
int main(void) {
 float a,b,res1;
 int res2,flag=1;
 char m;
 printf("请输入两个操作数和一个双目算数运算符:\n");
 scanf("%f%f%c", &a, &b, &m);
 switch (m){
 case '+':
  res1=a+b;
  break;
  case '-':
  res1 = a-b;
  break;
  case '*':
  res1 = a * b;
  break;
  case '/':
  res1 = a/b;
  break;
  case '%':
  res2 = (int) a % (int) b;
  flag=0;
  }
  if (flag)
  printf("%f\n", res1);
  else
  printf("%d\n", res2);

  return 0;
 }
```

3、

```
#include <stdio.h>
#include <stdlib.h>

```

```
/**
 * 验证列最小
 * @param m 二维数组头地址
 * @param i 需验证数所在行数
 * @param j 需验证数所在列数
 * @param n 阶数
  * @return
  */
 int verify(int *m,int i,int j,int n){
  int t = *(m+i*n+j);
  int flag=0;
  for (int k = 0; k < n; ++k) {
  if (*(m+k*n+j)<=t && k!=i)
  return flag;
  }
  flag=1;
  return flag;
 }
 int main(void) {
  int n;
  printf("请输入方阵阶数:\n");
  scanf("%d", &n);
  int *m = (int *) malloc(sizeof(int) * n * n);
  printf("请输入方阵中的元素，由左至右，由上至下:\n");
  for (int i = 0; i < n*n; ++i) {
  scanf("%d", m + i);
  }

  //寻找每行中的行最小值
  int j=0,min=0;
  for (int k = 0; k < n; ++k) {
  min = *(m + k * n);
  j=0;
  for (int l = 1; l < n; ++l) {
  if(*(m+k*n+l)<min){
  min = *(m+k*n+l);
  j=l;
  }
  }
  //验证行最小是否为列最小
```

```c
  if (verify(m, k, j, n)) {
  printf("%d(%d, %d)\n", min, k+1, j+1);
  }
  }

  return 0;
}
```