# Exercises 1

1-1 What are data, data element, atomic element? What are their differences?

1-2 What are data type, atomic data type, structure data type, abstract data type, virtual data type, intrinsic data type? What are the relationships among them?

1-3 What is the relationship between data structure and software? Solving practical problems, what is the general rules and when selecting (design) data structures?

1-4 How to understand the data (or abstract data) type and their instances? What is the value range of char and boolean data types in C++ language? What operations are defined?

1-5 What is the relationship between abstract data types (ADT) and the object-oriented approach? How to write ADT description? What are the advantages of using ADT?

1-6 Why do we say the logical relationships between data elements are the main aspects of the internal organization of data? What is random access, sequential access, direct access?

1-7 What is the relationship between logical structure and storage structure? What is the difference?

1-8 What is the relationship between operation and operation implementation? What are the similarities and differences? What is an operation?

1-9 Describe the distinction between the concept of data structure and concept of data type in programming language, the differences between C++ and Visual C++.

1-10 Give an example of data structure, describes its contents of logical structure, storage structure and operation.

1-11 What are the difference between algorithms and programs? What do we lead to the concept of an algorithm? How to evaluate the time and space complexity of the algorithm and if the algorithm is good or bad?

1-12 Analyze the following time complexity of the segments.

(1)the first segment

```
void odd (int n)
{   int i, j, x=0, y=0;
      for (i=1; i<=n; i++)
         if odd (i) {    for (j=i; j<=n; j++) x++;
                          for (j=1; j<=i; j++) y++;
```

```
                              }
          }
```

(2)the second segment

```
          void   recursive (int n)
          {   if (n<=1) return 1; else return (recursive(n–1)+recursive(n–1)); }
```

1-13 Let n be a positive integer, determine the frequency of the underlined statement in the following block.

①int i=1, k=0; while (i<=n–1) {   k=k+10*i;   i=i+1;   }

②int i=1, j=0; while ((i+j)<=n)   if (i>j)   j++;   else i++;

③int x=91, y=100; while (y>0) {   if (x>100) {   x=x–10;   y—;   }   else x++;   }

④int x=0;

```
     for (int i=1; i<=n; i++)
          for (int j=1; j<=i; j++)
               for (int k=1; k<=j; k++)   x++;
```

⑤int x=n, y=0;   //n>1

```
     while (x>=(y+1)*(y+1))   y++;
```

1-14 Compare the following four kinds of binary search program, which is correct? Which is more efficient? Assume the following variables and constants n>0.

Program A:

```
          void binsearchA (int a[n], int x, int &k)
          {   int i=0, j=n–1;
               while (i<=j)
               {   k= (i+j)/2; if (a[k]==x) return; if (x>a[k]) i=k+1; else j=k–1;   }
          }
```

Program B:

```
          void binsearchB (int a[n], int x, int &k)
          {   int i=0, j=n–1;
               while (i<j) {   k= (i+j)/2; if (a[k]==x) return; if (x>a[k])i=k; else j=k;   }
          }
```

Program C:

```
        void binsearchC (int a[n], int x, int &k)
        {   int i=0, j=n–1;
            while (i<=j) {   k= (i+j)/2; if (x<=a[k])j=k–1; if (x>=a[k] i=k+1;   }
        }
```

Program D:

```
        void binsearchd (int a[n], int x, int &k)
        {   int i=0, j=n–1;
            while (i<j) {   k= (i+j)/2; if (x<a[k]) j=k; else i=k+1;   }
        }
```

(**Hint**: if the element that we find is existent, then the program must be terminated at a[*k*]=x; If it is not existent, the program must be terminated at a[*k*]≠*x*.)