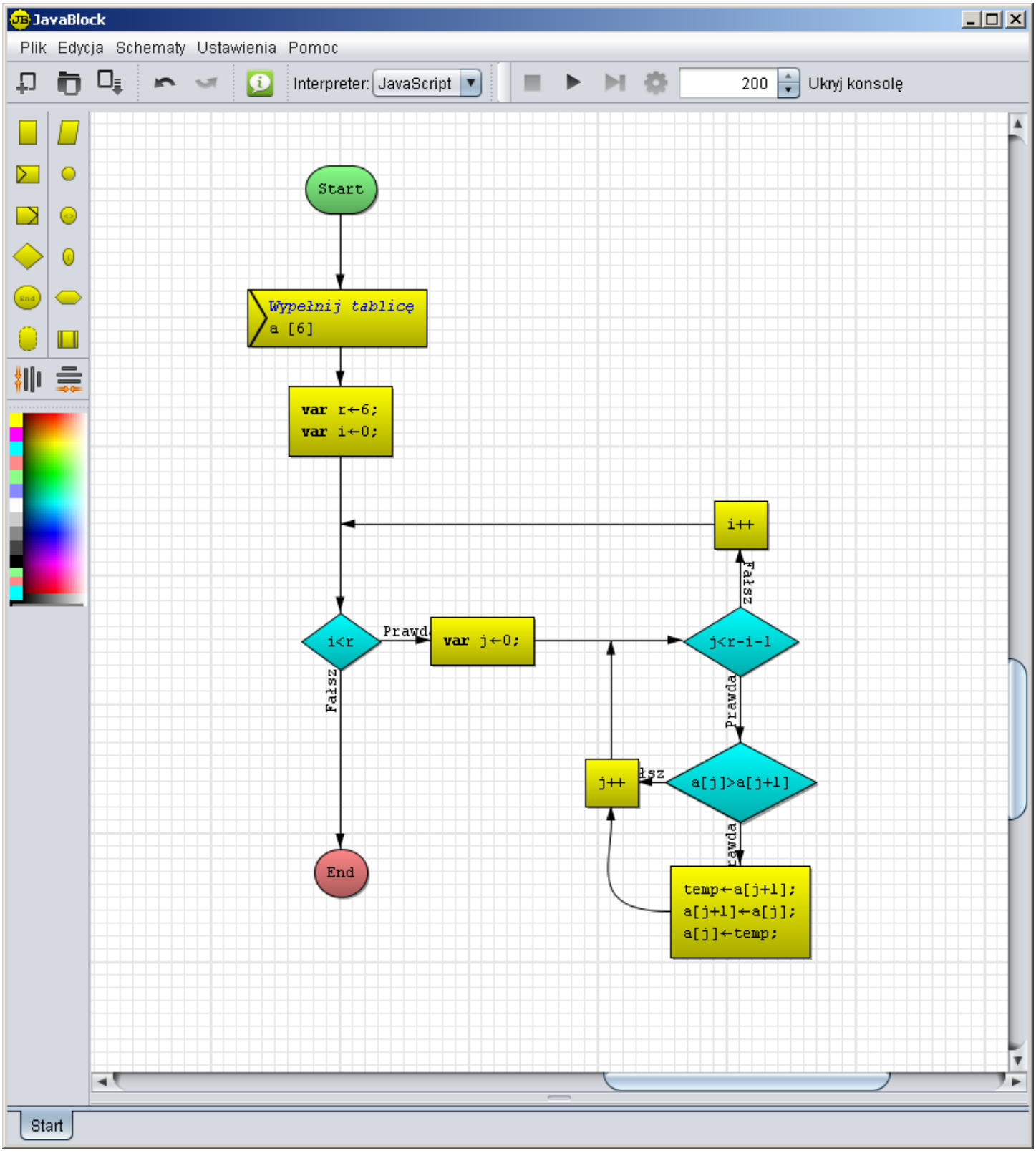
1. JavaBlock & C++

Task 1 (C++) Bubble sort. An array of sample elements: a={3, 7, 1, 4, 2, 8}.

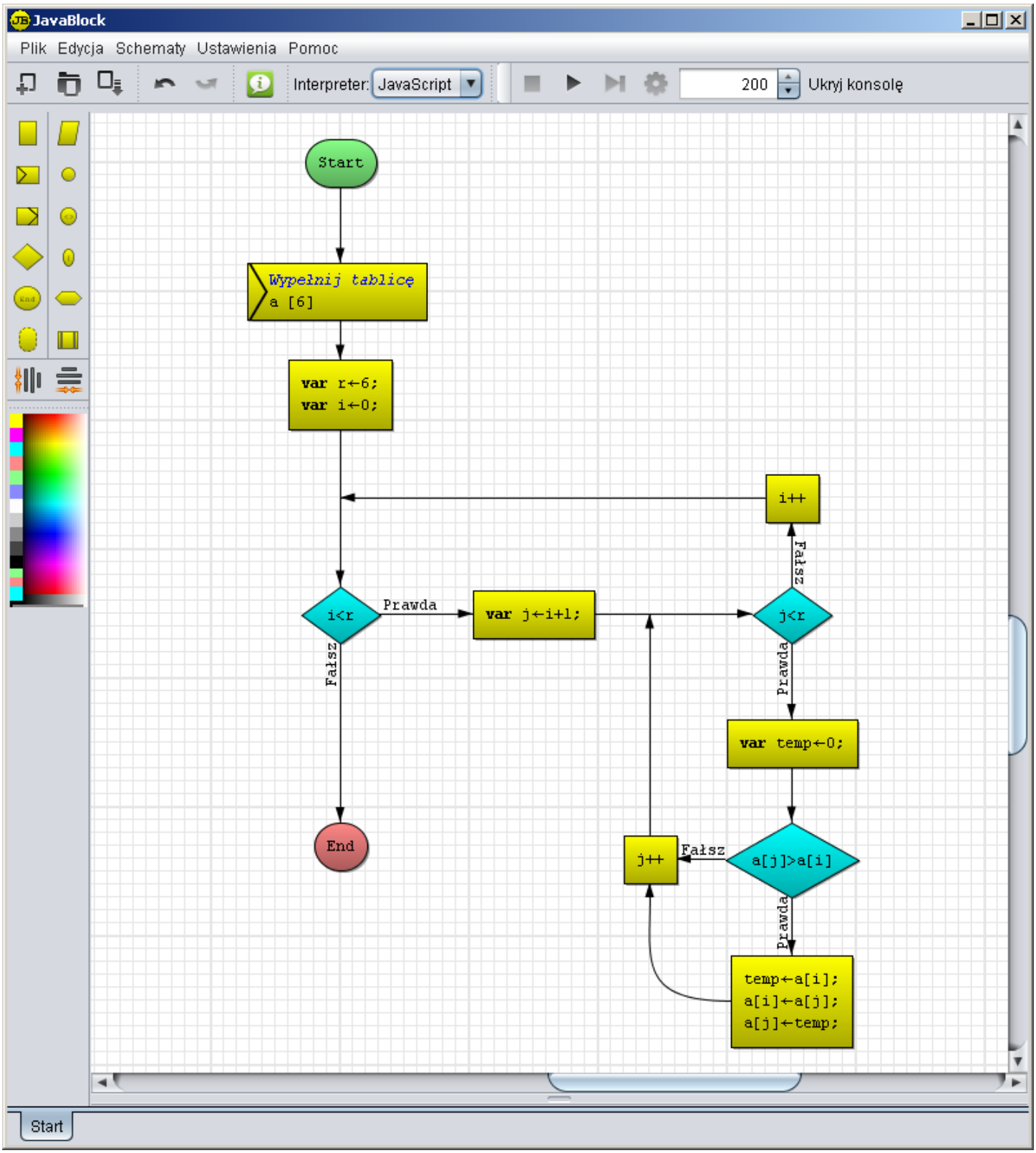
[t1.cpp]

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main(){

    int r=0;
    cout<<" Specify the size of an array:"<<endl;
    cin>>r;

    int *a=new int[r];

    cout<<"Enter the elements of the array:"<<endl;
    for(int i=0;i<r;i++)
        cin>>a[i];


    cout<<" An array before sorting:"<<endl;
        for(int i=0;i<r;i++)
        cout<<a[i]<<" ";

    cout<<endl;

    for(int i=0;i<r;i++)
    {
        cout<<endl;
        cout<<"i="<<i<<":";
        for(int j=0;j<r-i-1;j++)
        {
            if(a[j]>a[j+1])
            {
                int temp=a[j+1];
                a[j+1]=a[j];
                a[j]=temp;
            }

            // The intermediate stages of sorting:
            cout<<endl;
            for(int j=0;j<r;j++)
                cout<<a[j]<<" ";
        }
    }

    cout<<"\n An array after sorting:"<<endl;
    for(int i=0;i<r;i++)
        cout<<a[i]<<" ";

    delete [] a;
    //system("pause");
    return 0;
}
```

Task 2 (JavaBlock) Bubble sort.



Task 2a (JavaBlock) A variant of bubble sort.



Task 3 (JavaBlock) Create with using JavaBlock a block scheme (a flowchart) of an algorithm which can be used to obtain values of Fibonacci sequence.
Tip:

fib[i] = fib[i-1] + fib[i-2]

Task 4 (JavaBlock) Create with using JavaBlock a block scheme of an algorithm which can be used to find roots (zeros) of a quadratic function.
Tip:

$$f(x) = a\,x^2 + b\,x + c,$$

try (with using JavaBlock) to find all roots of polynomial:

$$x^2 - 5x + 6x == (x-2)(x-3). \quad (1)$$

Good idea is to provide a values for the parameters which you will be using in your program:

$a = 1, b = -5, c = 6.$

As an output of your program you should generate values of two roots (for equation (1)):

$x_1 = 2, x_2 = 3.$

Task 5 (JavaBlock) Use JavaBlock to create a block diagram of an algorithm that calculates the square root of any positive number (the so-called Babylonian method).
Tip (pseudocode):

```
x_{0}:=~\sqrt{S}
epsilon=0.001
    x_{1}:=(x_{0}+S/x_{0})/2
while abs(x_{n+1}-x_{n})>epsilon
    x_{n+1}:=(x_{n}+S/x_{n})/2
return x_{n+1}
```

Task 6 (JavaBlock) Using a JavaBlock program, create a block diagram of an algorithm whose we could use to determine whether a given number is prime.

Andrzej Pisarski