

## Sprawozdanie

### Zadanie 2

*Zadanie brzmi «Posortować ciąg liczb wykorzystując oraz algorytm scalania dwóch posortowanych ciągów. Zadany ciąg podzielono na dwie części. Każda z nich jest sortowana oddzielnie, następnie posortowane części są scalane. Sortowanie obu połówek oraz procedura scalająca wykonywane mają być współbieżnie.»*

Kod źródłowy rozwiązania:

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

short arr[] = {5, 6, 1, 4, 9, 2, 7, 8, 3, 0};

struct Attrs {
    short l;
    short m;
    short r;
};

void *merge(void *a) {
    struct Attrs *args = a;

    int i, j, k;
    int n1 = args->m - args->l + 1;
    int n2 = args->r - args->m;

    int L[n1], R[n2];

    for (i = 0; i < n1; ++i) {
        L[i] = arr[args->l + i];
    }

    for (j = 0; j < n2; ++j) {
        R[j] = arr[args->m + 1 + j];
    }

    i = 0;
    j = 0;
    k = args->l;
```

```

while (i < n1 && j < n2) {
    if (L[i] <= R[j]) {
        arr[k] = L[i];
        i++;
    } else {
        arr[k] = R[j];
        j++;
    }

    k++;
}

while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
}

while (j < n2) {
    arr[k] = R[j];
    j++;
    k++;
}

return NULL;
}

void *mergeSort(void *a) {
    struct Attrs *args = a;

    if (args->l < args->r) {
        pthread_t pid[3];
        struct Attrs attrs[3];

        int m = (args->l + args->r) / 2;

        attrs[0].l = args->l;
        attrs[0].r = m;
        pthread_create(&pid[0], NULL, mergeSort, &attrs[0]);

        attrs[1].l = m + 1;
        attrs[1].r = args->r;
        pthread_create(&pid[1], NULL, mergeSort, &attrs[1]);

        pthread_join(pid[0], NULL);
    }
}

```

```

        pthread_join(pid[1], NULL);

        attrs[2].l = args->l;
        attrs[2].m = m;
        attrs[2].r = args->r;
        pthread_create(&pid[2], NULL, merge, &attrs[2]);

        pthread_join(pid[2], NULL);
    }

    return NULL;
}

void printArray(short A[], short length) {
    printf("Array:\n");

    for (short i = 0; i < length; ++i) {
        printf("%d ", A[i]);
    }

    printf("\n");
}

int main() {
    short length = sizeof(arr) / sizeof(short);
    struct Attrs attrs;

    printArray(arr, length);

    attrs.l = 0;
    attrs.r = length - 1;
    mergeSort(&attrs);

    printArray(arr, length);

    return 0;
}

```

Wydruk programu:

Array:

5 6 1 4 9 2 7 8 3 0

Array:

0 1 2 3 4 5 6 7 8 9

### Zadanie 3

Zadanie brzmi «Napisać program znajdujący liczby pierwsze używając tzw. sita Erastotenesa»

Kod źródłowy rozwiązania:

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

#define LENGTH 100

short arr[LENGTH];

struct Attrs {
    short i;
};

void *sieve(void *a) {
    struct Attrs *args = a;

    for (short i = 0; i < LENGTH; ++i) {
        if (arr[args->i] >= 2 && arr[i] != arr[args->i] && arr[i]
% arr[args->i] == 0) {
            arr[i] = -1;
        }
    }
}

void printArray(short A[], short length) {
    printf("Array:\n");

    for (short i = 0; i < length; ++i) {
        if (A[i] >= 0) {
            printf("%d ", A[i]);
        }
    }
}
```

```

    printf("\n");
}

int main() {
    pthread_t pid[LENGTH];
    struct Attrs attrs[LENGTH];

    for (short i = 0; i < LENGTH; ++i) {
        arr[i] = i;
    }

    printArray(arr, LENGTH);

    for (short i = 1; i < LENGTH; ++i) {
        attrs[i].i = i;
        pthread_create(&pid[i], NULL, sieve, &attrs[i]);
    }

    for (short i = 1; i < LENGTH; ++i) {
        pthread_join(pid[i], NULL);
    }

    printArray(arr, LENGTH);

    return 0;
}

```

Wydruk programu:

```

Array:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87
88 89 90 91 92 93 94 95 96 97 98 99
Array:
1 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79
83 89 97

```