Memory allocation

#include <stdio.h>

typedef struct

{

    int size;

    int isFree;

} MemoryBlock;

int firstFit(MemoryBlock blocks[], int numBlocks, int processSize)

{

    for (int i = 0; i < numBlocks; i++)

    {

        if (blocks[i].isFree && blocks[i].size >= processSize)

        {

            blocks[i].isFree = 0;

            printf("Process of size %d allocated in block of size %d\n", processSize, blocks[i].size);

            return i;

        }

    }

    printf("No suitable block found for process of size %d\n", processSize);

    return -1;

}

int bestFit(MemoryBlock blocks[], int numBlocks, int processSize)

{

    int bestIndex = -1;

    for (int i = 0; i < numBlocks; i++)

    {

        if (blocks[i].isFree && blocks[i].size >= processSize)

        {

            if (bestIndex == -1 || blocks[i].size < blocks[bestIndex].size)

            {

                bestIndex = i;

            }

        }

    }

    if (bestIndex != -1)

    {

        blocks[bestIndex].isFree = 0;

        printf("Process of size %d allocated in block of size %d\n", processSize, blocks[bestIndex].size);

    }

    else

    {

        printf("No suitable block found for process of size %d\n", processSize);

    }

    return bestIndex;

}

int worstFit(MemoryBlock blocks[], int numBlocks, int processSize)

{

    int worstIndex = -1;

    for (int i = 0; i < numBlocks; i++)

    {

        if (blocks[i].isFree && blocks[i].size >= processSize)

        {

            if (worstIndex == -1 || blocks[i].size > blocks[worstIndex].size)

            {

                worstIndex = i;

            }

        }

    }

    if (worstIndex != -1)

    {

        blocks[worstIndex].isFree = 0;

        printf("Process of size %d allocated in block of size %d\n", processSize, blocks[worstIndex].size);

    }

    else

    {

        printf("No suitable block found for process of size %d\n", processSize);

    }

    return worstIndex;

}

int main()

{

    int numBlocks, choice, processSize;

    printf("Enter number of memory blocks: ");

    scanf("%d", &numBlocks);

    MemoryBlock blocks[numBlocks];

    for (int i = 0; i < numBlocks; i++)

    {

        printf("Enter size of block %d: ", i + 1);

        scanf("%d", &blocks[i].size);

        blocks[i].isFree = 1;

    }

    printf("Choose allocation method:\n1. First Fit\n2. Best Fit\n3. Worst Fit\n");

    scanf("%d", &choice);

    while (1)

    {

        printf("\nEnter the size of the process to allocate (0 to exit): ");

        scanf("%d", &processSize);

        if (processSize == 0)

            break;

        switch (choice)

        {

        case 1:

            firstFit(blocks, numBlocks, processSize);

            break;

        case 2:

            bestFit(blocks, numBlocks, processSize);

            break;

        case 3:

            worstFit(blocks, numBlocks, processSize);

            break;

        default:

            printf("Invalid choice!\n");

        }

    }

    printf("\nRemaining memory blocks:\n");

    for (int i = 0; i < numBlocks; i++)

    {

        printf("Block %d: Size = %d, Status = %s\n", i + 1, blocks[i].size, blocks[i].isFree ? "Free" : "Allocated");

    }

    return 0;

}