Лабораторная работа 9.

«Указатели, арифметика указателей. Введение в функции.»

Задание 1.

1. Напишите программу из нескольких файлов (модулей), включая файл основной программы. Файлы должны содержать вынесенные отдельно функции для выделения памяти под динамические двумерные и одномерные массивы и функции для перемножения матриц. Собрать проект используя утилиту Make.
2. Текст программы:

|  |  |
| --- | --- |
| **Makefile**  CC=gcc CFLAGS=-std=c17 DEPENDENCIES=mmult.h memfree.h memalloc.h print.h OBJECT\_DEPENDENCIES=main.o mmult.o memfree.o memalloc.o print.o LIBRARIES=-lm .DEFAULT\_GOAL=main  %.o: *%*.c $(DEPENDENCIES)  $(CC) $(CFLAGS) -c -o $@ $<  main:$(OBJECT\_DEPENDENCIES)  $(CC) $(CFLAGS) -o $@ $^ $(LIBRARIES) | **main.c**  #include <stdio.h> #include <stdlib.h> #include "memalloc.h" #include "mmult.h" #include "memfree.h" #include "print.h"  int main() {  int m1, n1;  int \*\*A;  printf("Enter sizes of first matrix in 'm n' - format: ");  if (scanf("%d %d", &m1, &n1)){  A = createMatrix(m1, n1);  }  else return -1;   int m2, n2;  int\*\* B;  printf("Enter sizes of second matrix in 'm n' - format: ");  if (scanf("%d %d", &m2, &n2)){  B = createMatrix(m2, n2);  }  else return -1;   int\*\* AmB = mmult(A, m1, n1, B, m2, n2);  printM(AmB, m1, n2);   int n;  int\* a;  printf("Enter size of array: ");  if (scanf("%d", &n)){  a = createArray(n);  }  else return -1;   printA(a, n);   freeM(A, m1);  freeM(B, m2);  freeM(AmB, m1);  freeA(a);  return 0; } |
| **memalloc.h**  #define MEMALLOC\_H  int\*\* createMatrix(int, int); int\* createArray(int); | **memalloc.c**  #include <stdio.h> #include <stdlib.h> #include "memalloc.h"  int\*\* createMatrix(int m, int n) {  int \*\*M = (int\*\*)malloc(n \* sizeof(int\*));  for (int i = 0; i < m; i++)  M[i] = (int\*)malloc(n \* sizeof(int));  for (int i = 0; i < m; i++)  for (int j = 0; j < n; j++) {  printf("[%d][%d] = ", i, j);  scanf("%d", &M[i][j]);  }  return M; } int\* createArray(int n) {  int \*A = (int\*)malloc(n \* sizeof(int));  for (int i = 0; i < n; i++)  {  printf("[%d] = ", i);  scanf("%d", &A[i]);  }  return A; } |
| **memfree.h**  #define MEMFREE\_H  void freeM(int\*\*, int); void freeA(int\*); | **memfree.c**  #include <stdio.h> #include <stdlib.h> #include "memfree.h"  void freeM(int \*\*M, int m) {  for (int i = 0; i < m; i++)  free(M[i]);  free(M); }  void freeA(int \*A) {  free(A); } |
| **mmult.h**  #define MMULT\_H  int\*\* mmult(int\*\*, int, int, int\*\*, int, int); | **mmult.c**  #include <stdio.h> #include <stdlib.h> #include "mmult.h"  int\*\* mmult(int\*\* M1, int m1, int n1, int\*\* M2, int m2, int n2) {  if (n1 == m2)  {  int\*\* M = (int\*\*)malloc(m1 \* sizeof(int\*));  for (int i = 0; i < m1; i++)  M[i] = (int\*)malloc(n2 \* sizeof(int));  for (int i = 0; i < m1; i++)  for (int j = 0; j < n2; j++) {  M[i][j] = 0;  for (int r = 0; r < n1; r++)  M[i][j] += M1[i][r] \* M2[r][j];  }  return M;  } else {  printf("Multplication isn't possible.");  return -1;  } } |
| **print.c**  #define PRINT\_H  void printM(int\*\*, int, int); void printA(int\*, int); | **print.h**  #include <stdio.h> #include <stdlib.h> #include "print.h"  void printM(int \*\*M, int m, int n) {  printf("Multiplication:\n");  for (int i = 0; i < m; i++){  for (int j = 0; j < n; j++) {  printf("[%d][%d] = %d\t", i, j, M[i][j]);  }  printf("\n");  } }  void printA(int \*A, int n) {  printf("Array:\n");  for (int i = 0; i < n; i++) {  printf("[%d] = %d\t", i, A[i]);  } } |

1. Результат выполнения:

