|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБАРАТОРНОЙ РАБОТЫ №5**

по дисциплине: «Операционные системы»

|  |  |  |
| --- | --- | --- |
| Студент |  | Беркович Евгений Сергеевич |
| Группа |  | РК6-55Б |
| Тип задания |  | Игра с компьютером |
| Вариант |  | 02 |

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_ **Беркович Е. С.**\_ \_

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_ **Грошев С. В.** \_\_

*подпись, дата фамилия, и.о.*

*Москва, 2023 г.*

**Техническое задание**

Требуется разработать программу, которая имитирует игру в кости. Играет компьютер и человек, выполняя последовательно 2 броска. Результат оценивается по сумме числа очков в обеих попытках. В качестве параметра командной строки вводится интервал времени в секундах, в течение которых компьютер генерирует случайные числа – состояние куба на экране, индицируемое целыми числами от 1 до 6.

**Приложение №1**

**Содержимое файла popular.c**

|  |
| --- |
| #include <unistd.h>  #include <stdlib.h>  #include <signal.h>  #include <setjmp.h>  #include <string.h>  typedef void (\*sighandler)(int);  void brake(int);  sigjmp\_buf pos;  static char digit[3];  static int j;  static char\* who[] = { "Man: ", "Man: ", "Com: ", "Com: ", "Bye.\n" };  static int man\_res = 0, com\_res = 0;  char\* result\_to\_string(int res, int \*num\_size);  char\* rewrite\_1(char\* str, size\_t size);  char\* top(int res1, int res2, int \*size\_res);  int main(int argc, char\*\* argv) {  char\* result = NULL;  int timeout = 0, size\_win\_result = 0;  j = 0;  if (argc < 2) {  write(2, "Usage: dice timeout\n", 20);  exit(0);  }  timeout = atoi(argv[1]);  if (timeout < 1) {  timeout = 1;  }  digit[0] = '1' + getpid() % 6;  digit[1] = '\b';  digit[2] = '\n';  j = sigsetjmp(pos, SIGINT); // сохраняется состояние SIGINT  signal(SIGALRM, (sighandler) brake);  signal(SIGINT, (sighandler) brake);  if (j == 2 || j == 3) {  alarm(timeout);  write(1, digit, 2);  }  else {  alarm(0);  }  if (j < 4) {  write(1, who[j], 5);  while (j < 4) {  write(1, digit, 2);  digit[0] = (digit[0] < '6') ? digit[0] + 1 : '1';  usleep(50000);  }  } else {  result = top(man\_res, com\_res, &size\_win\_result);  if (result == NULL) {  write(0, "Memory error\n", 13);  exit(0);  }  write(0, result, size\_win\_result);  write(0, "\n", 1);  write(1, who[j], 5);  }  signal(SIGINT, SIG\_DFL);  exit(0);  }  void brake(int c) {  if (j < 2) {  man\_res += digit[0] - '0';  } else {  com\_res += digit[0] - '0';  }  write(1, digit, 3);  siglongjmp(pos, ++j);  return;  }  char\* top(int res1, int res2, int \*size\_res) {  char\* res = NULL;  if (res1 > res2) {  return result\_to\_string(res1, size\_res);  } else if (res2 > res1) {  return result\_to\_string(res2, size\_res);  } else {  res = (char\*)malloc(5 \* sizeof(char));  strcpy(res, "draw");  \*size\_res = 4;  return res;  }  }  char\* rewrite\_1(char\* str, size\_t size) {  char\* new\_str = (char\*)malloc((size + 1) \* sizeof(char));  size\_t i;  for (i = size; i > 0; i--) {  new\_str[i] = str[i - 1];  }  free(str);  return new\_str;  }  char\* result\_to\_string(int res, int \*num\_size) {  char \*str = (char\*)malloc(sizeof(char));  size\_t i = 1;  while (res / 10 != 0) {  str[0] = (res % 10) + '0';  str = rewrite\_1(str, i);  i++;  res/=10;  \*num\_size += 1;  }  str[0] = res + '0';  \*num\_size += 1;  return str;  } |

**Содержимое файла text.txt**

This is an example of text that includes multiple repeated Words. Repeated Words can make text quite interesting. Can you identify which words are repeated? I bet you can! You can use the program we discussed to count how often individual words are used. This could be a fun learning exercise. Exercise your programming skills and it will benefit you in future learning endeavours. So, isn't this a good way to learn more about word frequency analysis?

**Приложение №2**

**Результат работы программы:**

