Контрольная работа

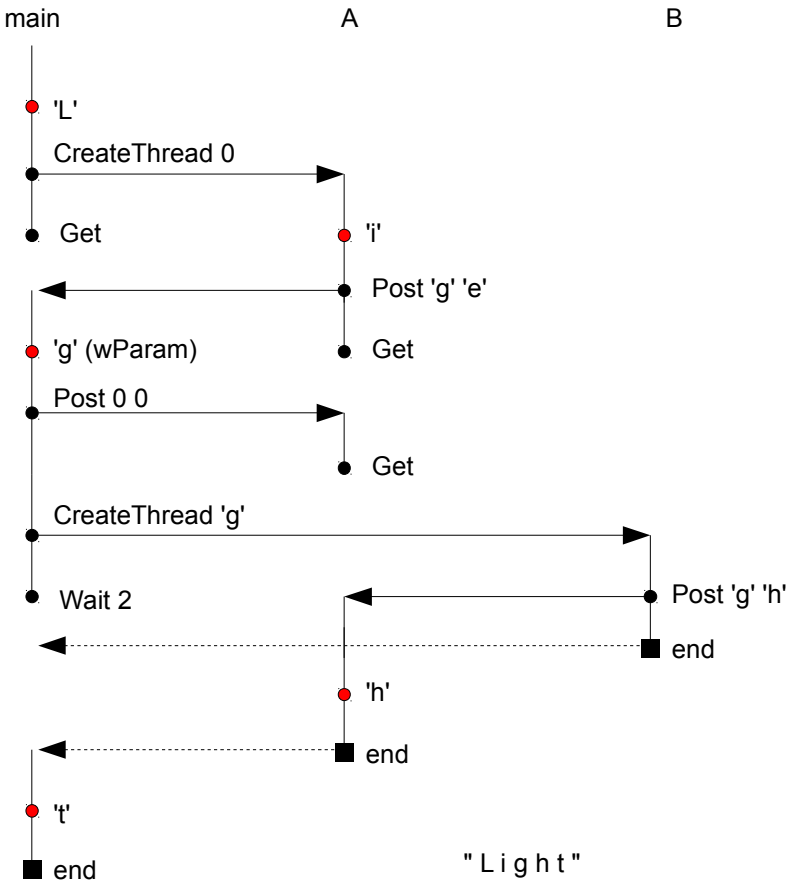
Задание. Дана программа (вариант определяется по последней цифре номера зачетки). Необходимо проанализировать работу программы и объяснить порядок, в котором выводятся символы (вывод символов осуществляется функцией putchar). Отчет представить в виде pdf файла в ЭИОС.

Образец решения контрольной работы

Программа:

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

unsigned long idM, idA, idB;

void A ()
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  putchar ('i');
  PostThreadMessage (idM, WM_USER, 'g', 'e');
  GetMessage (&msg, 0, 0, 0);
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  }

void B (char c)
  {
  PostThreadMessage (idA, WM_USER, c, c + 1);
  }

int main ()
  {
  HANDLE TH [2];
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  idM = GetCurrentThreadId ();
  putchar ('L');
  TH[0] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)A, 0, 0, &idA);
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.wParam);
  PostThreadMessage (idA, WM_USER, 0, 0);
  TH[1] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)B, (LPVOID) 'g', 0, &idB);
  WaitForMultipleObjects (2, TH, TRUE, INFINITE);
  putchar ('t');
  return 0;
  }
```

Блок-схема работы программы с акцентом на функции, определяющие порядок выводимых символов и сами эти символы, представлена на рисунке ниже.



Объяснение. Программа начинает выполнение с функции main. PeekMessage вызывается для того, чтобы система создала очередь сообщений для нити main. Идентификатор нити main получается и запоминается в глобальной переменной idM. Печатается первая буква L. Создается нить A с нулевым параметром (CreateThread). Обе нити работают параллельно, но main блокируется на приеме сообщения (GetMessage). Нить A печатает i и посылает сообщение g e нити main. После этого A блокируется на приеме сообщения, а main выходит из блокировки (из ожидания поступления сообщения) и печатает g (из параметра wParam сообщения). После этого main посылает нити A пустое сообщение и начинает создавать нить B (CreateThread с параметром g). Одновременно с этим A вызывает GetMessage и блокируется до прихода сообщения. Нити main и B продолжают работать параллельно, но main вызывает функцию ожидания завершения нитей A и B (WaitForMultipleObjects). Нить B посылает нити A сообщение g h и завершается. Нить A выходит из блокировки, печатает h (из lParam) и завершается. Завершение второй нити пробуждает main, main печатает t и завершает весь процесс. Итоговое напечатанное слово – “Light”.

Далее, каждая на своей странице, идут программы по вариантам.

Вариант 0

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

unsigned long idM, idA, idB;

void A ()
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'p', 'q');
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  }

void B (char c)
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'o', c);
  PostThreadMessage (idA, WM_USER, 'y', 'o');
  }

int main ()
  {
  HANDLE TH [2];
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  idM = GetCurrentThreadId ();
  putchar ('M');
  TH[0] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)A, 0, 0, &idA);
  GetMessage (&msg, 0, 0, 0);
  TH[1] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)B, (LPVOID) 'n', 0, &idB);
  GetMessage (&msg, 0, 0, 0);
  WaitForMultipleObjects (2, TH, TRUE, INFINITE);
  putchar (msg.wParam);
  putchar (msg.lParam);
  return 0;
  }
```

Вариант 1

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

unsigned long idM, idA, idB;

void A (char c)
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'c', 'b');
  putchar ('d');
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  }

void B (char c)
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'e', c);
  PostThreadMessage (idA, WM_USER, 'i', 'w');
  }

int main ()
  {
  HANDLE TH [2];
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  idM = GetCurrentThreadId ();
  TH[0] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)A, (LPVOID) 'o', 0, &idA);
  GetMessage (&msg, 0, 0, 0);
  TH[1] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)B, (LPVOID) 'n', 0, &idB);
  GetMessage (&msg, 0, 0, 0);
  putchar ('a');
  WaitForMultipleObjects (2, TH, TRUE, INFINITE);
  putchar (msg.lParam);
  return 0;
  }
```

Вариант 2

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

unsigned long idM, idA, idB;

void A (char c)
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  putchar ('E');
  PostThreadMessage (idM, WM_USER, 'c', 'a');
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  }

void B (char c)
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'h', c);
  PostThreadMessage (idA, WM_USER, 'p', 't');
  putchar ('r');
  }

int main ()
  {
  HANDLE TH [2];
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  idM = GetCurrentThreadId ();
  TH[0] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)A, (LPVOID) 'q', 0, &idA);
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  TH[1] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)B, (LPVOID) 'm', 0, &idB);
  GetMessage (&msg, 0, 0, 0);
  WaitForMultipleObjects (2, TH, TRUE, INFINITE);
  putchar (msg.wParam);
  return 0;
  }
```

Вариант 3

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

unsigned long idM, idA, idB;

void A ()
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  putchar ('h');
  PostThreadMessage (idM, WM_USER, 'o', 'e');
  GetMessage (&msg, 0, 0, 0);
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  }

void B (char c)
  {
  PostThreadMessage (idA, WM_USER, c, c - 1);
  }

int main ()
  {
  HANDLE TH [2];
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  idM = GetCurrentThreadId ();
  TH[0] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)A, 0, 0, &idA);
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.wParam);
  PostThreadMessage (idA, WM_USER, 0, 0);
  putchar ('u');
  TH[1] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)B, (LPVOID) 't', 0, &idB);
  WaitForMultipleObjects (2, TH, TRUE, INFINITE);
  putchar (msg.lParam);
  return 0;
  }
```

Вариант 4

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

unsigned long idM, idA, idB;

void A ()
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'r', 'n');
  GetMessage (&msg, 0, 0, 0);
  putchar ('s');
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  }

void B (char c)
  {
  PostThreadMessage (idA, WM_USER, c + 1, c);
  putchar ('w');
  }

int main ()
  {
  HANDLE TH [2];
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  idM = GetCurrentThreadId ();
  putchar ('a');
  TH[0] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)A, 0, 0, &idA);
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  PostThreadMessage (idA, WM_USER, 0, 0);
  TH[1] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)B, (LPVOID) 'e', 0, &idB);
  WaitForMultipleObjects (2, TH, TRUE, INFINITE);
  putchar (msg.wParam);
  return 0;
  }
```

Вариант 5

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

unsigned long idM, idA, idB;

void A ()
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  putchar ('b');
  PostThreadMessage (idM, WM_USER, 'e', 'y');
  GetMessage (&msg, 0, 0, 0);
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  }

void B (char c)
  {
  PostThreadMessage (idA, WM_USER, c, c - 1);
  }

int main ()
  {
  HANDLE TH [2];
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  idM = GetCurrentThreadId ();
  TH[0] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)A, 0, 0, &idA);
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.wParam);
  PostThreadMessage (idA, WM_USER, 0, 0);
  putchar ('r');
  TH[1] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)B, (LPVOID) 's', 0, &idB);
  WaitForMultipleObjects (2, TH, TRUE, INFINITE);
  putchar (msg.lParam);
  return 0;
  }
```

Вариант 6

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

unsigned long idM, idA, idB;

void B (char c)
  {
  PostThreadMessage (idA, WM_USER, c + 1, c);
  putchar ('t');
  }

void A ()
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'r', 'e');
  GetMessage (&msg, 0, 0, 0);
  putchar ('n');
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  }

int main ()
  {
  HANDLE TH [2];
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  idM = GetCurrentThreadId ();
  putchar ('c');
  TH[0] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)A, 0, 0, &idA);
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  PostThreadMessage (idA, WM_USER, 0, 0);
  TH[1] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)B, (LPVOID) 'e', 0, &idB);
  WaitForMultipleObjects (2, TH, TRUE, INFINITE);
  putchar (msg.wParam);
  return 0;
  }
```

Вариант 7

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

unsigned long idM, idA, idB;

void A (char c)
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  putchar ('d');
  PostThreadMessage (idM, WM_USER, 'c', 'e');
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  }

void B (char c)
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'c', c);
  PostThreadMessage (idA, WM_USER, 'p', 'i');
  putchar ('v');
  }

int main ()
  {
  HANDLE TH [2];
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  idM = GetCurrentThreadId ();
  TH[0] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)A, (LPVOID) 'q', 0, &idA);
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  TH[1] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)B, (LPVOID) 'm', 0, &idB);
  GetMessage (&msg, 0, 0, 0);
  WaitForMultipleObjects (2, TH, TRUE, INFINITE);
  putchar (msg.wParam);
  putchar ('e');
  return 0;
  }
```

Вариант 8

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

unsigned long idM, idA, idB;

void A (char c)
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'r', 'q');
  putchar ('f');
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  }

void B (char c)
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'e', c);
  PostThreadMessage (idA, WM_USER, 'i', 'l');
  }

int main ()
  {
  HANDLE TH [2];
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  idM = GetCurrentThreadId ();
  TH[0] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)A, (LPVOID) 'o', 0, &idA);
  GetMessage (&msg, 0, 0, 0);
  TH[1] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)B, (LPVOID) 'm', 0, &idB);
  GetMessage (&msg, 0, 0, 0);
  putchar ('i');
  WaitForMultipleObjects (2, TH, TRUE, INFINITE);
  putchar (msg.lParam);
  return 0;
  }
```

Вариант 9

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

unsigned long idM, idA, idB;

void A ()
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'p', 'q');
  GetMessage (&msg, 0, 0, 0);
  putchar (msg.lParam);
  }

void B (char c)
  {
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  PostThreadMessage (idM, WM_USER, 'n', c);
  PostThreadMessage (idA, WM_USER, 'y', 'i');
  }

int main ()
  {
  HANDLE TH [2];
  MSG msg;
  PeekMessage (&msg, 0, 0, 0, PM_NOREMOVE);
  idM = GetCurrentThreadId ();
  putchar ('K');
  TH[0] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)A, 0, 0, &idA);
  GetMessage (&msg, 0, 0, 0);
  TH[1] = CreateThread (0, 0, (LPTHREAD_START_ROUTINE)B, (LPVOID) 'g', 0, &idB);
  GetMessage (&msg, 0, 0, 0);
  WaitForMultipleObjects (2, TH, TRUE, INFINITE);
  putchar (msg.wParam);
  putchar (msg.lParam);
  return 0;
  }
```