Выполнила: Белоусова Е., ИП-911

Задача

Цель: знакомство с работой компилятора Visual C++ cl в командной строке и знакомство с документацией Windows API в MSDN.

Упражнение 1. Программно определить пути к системному каталогу Windows и каталогу временных файлов Windows, используя следующие функции Win32 API:
• UINT GetWindowsDirectory(LPTSTR lpBuffer, UINT uSize);
• DWORD GetTempPath(DWORD nBufferLength, LPSTR lpBuffer);

Упражнение 2. Используя функции Windows API получить системную информацию: количество ядер процессора, архитектуру процессора, нижнюю и верхнюю границы пользовательского адресного пространства (ключевое слово для поиска в MSDN - GetSystemInfo).

Упражнение 3. Используя функции Windows API получить информацию об использовании физической и виртуальной памяти (ключевое слово для поиска в MSDN - GlobalMemoryStatus).

Примечание: использовать 32- и 64- разрядные версии компилятора.

Описание работы программы

Программно определим пути к системному каталогу Windows и каталогу временных файлов. Установим путь для компилятора:

```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\grant> cmd.exe /k "C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\VC\Auxiliary\Build\vcvars32.bat" `& powershell
**********************************************************************
** Visual Studio 2019 Developer Command Prompt v16.8.4
** Copyright (c) 2020 Microsoft Corporation
**********************************************************************
[vcvarsall.bat] Environment initialized for: 'x86'
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\grant> cd  D:\"семестр 5"\os\"лаб2"
PS D:\семестр 5\os\лаб2> cl osLab2.c
Оптимизирующий компилятор Microsoft (R) C/C++ версии 19.28.29336 для x86
(C) Корпорация Майкрософт (Microsoft Corporation).  Все права защищены.

osLab2.c
Microsoft (R) Incremental Linker Version 14.28.29336.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:osLab2.exe
osLab2.obj
```

Воспользуемся функциями GetWindowsDirectory и GetTempPath.

```
PS D:\семестр 5\os\лаб2> ./osLab2.exe
Temp path is C:\Users\grant\AppData\Local\Temp\
Windows directory path is C:\Windows
```

Получим системную информацию с помощью функции GetSystemInfo. Для этого создадим переменную типа SYSTEM_INFO. Выведем необходимые нам поля этой структуры:

dwNumberOfProcessors - количество логических процессоров.

wProcessorArchitecture – архитектура процессора. Возвращает значение, которое нужно сопоставить с таблицей из документации.

Value	Meaning
PROCESSOR_ARCHITECTURE_AMD64 9	x64 (AMD or Intel)
PROCESSOR_ARCHITECTURE_ARM 5	ARM
PROCESSOR_ARCHITECTURE_ARM64 12	ARM64
PROCESSOR_ARCHITECTURE_IA64 6	Intel Itanium-based
PROCESSOR_ARCHITECTURE_INTEL 0	x86
PROCESSOR_ARCHITECTURE_UNKNOWN 0xffff	Unknown architecture.

lpMinimumApplicationAddress - указатель на наименьший адрес памяти.

lpMaximumApplicationAddress – указатель на верхнюю границу адреса памяти.

```
Number Of Processors: 4
Processor Type: 0
Minimum Application Address: 00010000
Maximum Application Address: 7FFEFFFF
```

Получим информацию об использовании физической и виртуальной памяти с помощью функции GlobalMemoryStatus. Для этого создадим переменную типа MEMORYSTATUS. Понадобятся поля:

dwTotalPhys - объем физической памяти в байтах.

dwAvailPhys - объем физической памяти, доступной в данный момент, в байтах.

dwTotalVirtual - общий объем виртуальной памяти в байтах.

dwAvailVirtual - доступный объем виртуальной памяти в байтах.

Использовав 32-битный компилятор получили следующий результат:

```
Total Physical: 2.00 GB
Avail Physical: 2.00 GB
Total Virtual: 0.00 TB
Avail Virtual: 0.00 TB
```

Что не является верными значениями. Обратившись к документации получаем объяснение:

На компьютерах Intel x86 с объемом памяти более 2 ГБ и менее 4 ГБ функция GlobalMemoryStatus всегда возвращает 2 ГБ в элементе dwTotalPhys структуры MEMORYSTATUS. Аналогично, если общий объем доступной памяти составляет от 2 до 4 ГБ, элемент dwAvailPhys структуры MEMORYSTATUS будет округлен до 2 ГБ.

Воспользуемся 64-битным компилятором.

```
PS D:\семестр 5\os\лаб2> cmd.exe /k "C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\VC\Auxiliary\Build\vcvars64.bat" `& powershell
**********************************************************************
** Visual Studio 2019 Developer Command Prompt v16.8.4
** Copyright (c) 2020 Microsoft Corporation
**********************************************************************
[vcvarsall.bat] Environment initialized for: 'x64'
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS D:\семестр 5\os\лаб2> cl osLab2.c
Оптимизирующий компилятор Microsoft (R) C/C++ версии 19.28.29336 для x64
(C) Корпорация Майкрософт (Microsoft Corporation).  Все права защищены.

osLab2.c
Microsoft (R) Incremental Linker Version 14.28.29336.0
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:osLab2.exe
osLab2.obj
PS D:\семестр 5\os\лаб2> ./osLab2.exe
```

```
Total Physical: 7.70 GB
Avail Physical: 3.14 GB
Total Virtual: 128.00 TB
Avail Virtual: 127.9961 TB
```

Возвращает теоретически максимально возможный объем виртуальной памяти?

Листинг

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

int main()
{
   char Buffer[100];
   int size = sizeof(Buffer);
   if (!GetTempPath((DWORD)size, (LPTSTR)Buffer))
   {
      printf("System error code: %i\n", GetLastError());
      return -1;
   }
   fprintf(stdout,"Temp path is %s\n", Buffer);
   if (!GetWindowsDirectory((LPTSTR)Buffer, (DWORD)size))
   {
      printf("System error code: %i\n", GetLastError());
      return -1;
   }
   fprintf(stdout,"Windows directory path is %s\n", Buffer);

   SYSTEM_INFO sSysInfo;
         GetSystemInfo(&sSysInfo);
   fprintf(stdout, "Number Of Processors: %d\n",  sSysInfo.dwNumberOfProcessors);
```

```
    fprintf(stdout, "Processor Type: %d\n",  sSysInfo.wProcessorArchitecture);
    fprintf(stdout, "Minimum Application Address: %lp\n",  sSysInfo.lpMinimumApplicationAddress);
    fprintf(stdout, "Maximum Application Address: %lp\n",  sSysInfo.lpMaximumApplicationAddress);

    MEMORYSTATUS sMem;
    GlobalMemoryStatus(&sMem);
    fprintf(stdout, "Total Phys: %.3f GB\n", sMem.dwTotalPhys / (1024.0 * 1024.0 * 1024.0));
    fprintf(stdout, "Avail Phys: %.3f GB\n", sMem.dwAvailPhys / (1024.0 * 1024.0 * 1024.0));

    fprintf(stdout, "Total Virtual: %.3f TB\n", sMem.dwTotalVirtual / (1024.0 * 1024.0 * 1024.0 * 1024.0));
    fprintf(stdout, "Avail Virtual: %.3f TB\n", sMem.dwAvailVirtual / (1024.0 * 1024.0 * 1024.0 * 1024.0));
    return 0;
}
```