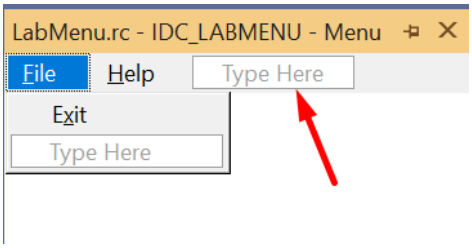
Lab #06

Работа с менюта. Обслужваща функция. Обработка на съобщения, манипулатори на опциите. Създаване, показване и модификация на менюта. Pop-up меню. Примерна програма за управление на меню.

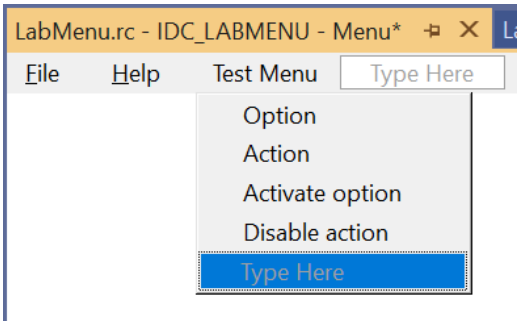
Solution/Project: нов

Раздел 1. Работа с елементите на основно меню

1. Създаваме нов проект LabMenu от тип C++/Windows/Desktop/Application (VS2019)
2. От Resource View отваряме меню IDC_LABMENU
3. Създаваме нов основен елемент като редактираме на посоченото място с червената стрелка



4. Добавяме елементи в меню „Test Menu“, за да получим следния изглед:



5. Променяме ID имената на меню елементите, както следва:
   a. Option: IDM_OPTION
   b. Action IDM_ACTION
   c. Activate option IDM_ACTIVATE_OPT
   d. Disable action IDM_DISABLE_ACT
6. Меню IDC_LABMENU е зададено да се използва в приложението във функция `MyRegisterClass` чрез:
   `wcex.lpszMenuName = MAKEINTRESOURCEW(IDC_LABMENU);`
7. Обслужващата функция на приложението е `WndProc` и в нея се обработват съобщенията от менюто в частта преди `case IDM_ABOUT`:

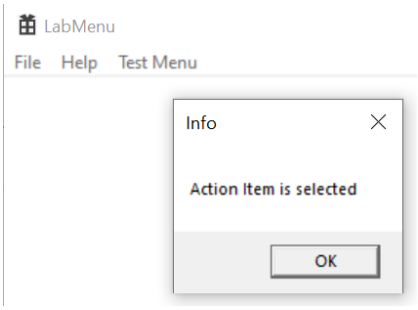
```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
    case WM_COMMAND:
        {
            int wmId = LOWORD(wParam);
            // Parse the menu selections:
            switch (wmId)
            {
              // here to insert message processing for menu items
            case IDM_ABOUT:
                DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
                    break;
```

8. Обработване на съобщение от меню елемент „Option“:

```
            switch (wmId)
            {
                    // here to insert message processing for menu items
                    case IDM_OPTION:
```

```
                        MessageBox(hWnd, L"Option Item is selected", L"Info", MB_OK);
                        break;
```

9. По аналогичен начин направете MessageBox и меню елемент „Action“



10. За следващите функционалности е необходимо да дефинирате следните променливи в началото на обслужващата функция на приложението е `WndProc`:

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        HMENU hMenu, hSubMenu;
        DWORD dwRes;
        BOOL bRes;
        UINT uRes;
        MENUITEMINFO miinf;
```

11. Съобщението от меню елемент „Activate option“ ще използваме да сложим чек-марк на меню елемент „Option“

```
                // here to insert message processing for menu items
                case IDM_ACTIVATE_OPT:
                        hMenu = GetMenu(hWnd);
                        dwRes = CheckMenuItem(hMenu, IDM_OPTION, MF_CHECKED);
                        break;
```

12. Съобщението от меню елемент „Disable action“ ще използваме да забраним използването на меню елемент „Action“

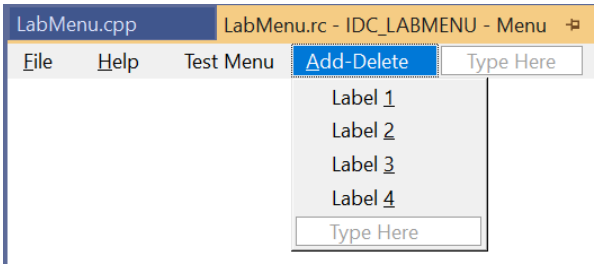
```
                // here to insert message processing for menu items
                case IDM_DISABLE_ACT:
                        hMenu = GetMenu(hWnd);
                        bRes = EnableMenuItem(hMenu, IDM_ACTION, MF_DISABLED);
                        break;
```

Проверете каква разлика има, ако вместо `MF_DISABLED` използвате `MF_GRAYED`

13. Нека коригираме действието на меню елемент с ID `IDM_DISABLE_ACT` така, че при всеки избор да променя състоянието на меню елемент „Action“ алтернативно на enabled/disabled и с подходяща промяна на текста за указване на очакваното действие

```
                // here to insert message processing for menu items
                case IDM_DISABLE_ACT:
                        hMenu = GetMenu(hWnd);
                        uRes = GetMenuState(hMenu, IDM_ACTION, MF_BYCOMMAND);
                        bRes = EnableMenuItem(hMenu, IDM_ACTION,
                                uRes & MF_DISABLED ? MF_ENABLED : MF_DISABLED);
                        bRes = ModifyMenu(hMenu, IDM_DISABLE_ACT, MF_BYCOMMAND,
IDM_DISABLE_ACT, uRes & MF_DISABLED ? L"Disable action" : L"Enable action");
                        break;
```

14. Създаваме нов основен елемент „Add-Delete” и елементи към него, за да получим следния изглед:



ID на елементите са IDM_ADD_LABEL1, IDM_ADD_LABEL2, IDM_ADD_LABEL3, IDM_ADD_LABEL4.

Обърнете внимание на знака “_” underscore на символите A, 1, 2, 3, 4. Този знак показва бърз клавиш за избор на меню елемент, който се задава чрез знак “&” пред съответния символ в Caption на меню елемента. Например „&Add-Delete”, „Label &3”. Бързият клавиш се активира с натискане и задържане на клавиш Alt, например Alt+A+4 за меню елемент “Label 4”.

15. Съобщението от меню елемент „Label 1“ ще добави нов меню елемент в края на списъка с елементи на “Add-Delete”

```
		// here to insert message processing for menu items
		case IDM_ADD_LABEL1:
			hMenu = GetMenu(hWnd);
			hSubMenu = GetSubMenu(hMenu, 3);
			if (GetMenuState(hMenu, IDM_ADD_LABEL4 + 1, MF_BYCOMMAND) == -1)
				bRes = AppendMenu(hSubMenu, MF_STRING, IDM_ADD_LABEL4 + 1,
L"New Menu Item");
			break;
		case IDM_ADD_LABEL4+1:
			MessageBox(hWnd, L"New Menu Item is selected", L"Info", MB_OK);
			break;
```

Функцията `GetSubMenu` ни връща подменюто „Add-Delete” като позиция 3 на основното меню („File” е 0).
Ако функцията `GetMenuState` върне -1, то меню елементът не съществува. Ако проверката липсва ще се добавят множество меню елемента с текст `"New Menu Item"`. Можете да проверите като сложите коментар на проверката `if (GetMenuState…`
Използвахме за ID на новия меню елемент увеличената с 1 стойност на ID на IDM_DISABLE_ACT. Затова и същата стойност използваме в `case` за обработка на съобщението за меню елемента `"New Menu Item"`.

16. Съобщението от меню елемент „Label 2“ ще изтрие новодобавения меню елемент `"New Menu Item"`.

```
		// here to insert message processing for menu items
		case IDM_ADD_LABEL2:
			hMenu = GetMenu(hWnd);
			bRes = DeleteMenu(hMenu, IDM_ADD_LABEL4 + 1, MF_BYCOMMAND);
			break;
```

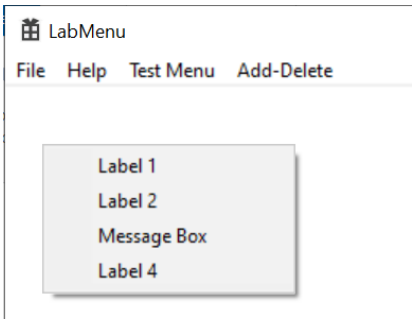
17. Съобщението от меню елемент „Label 4“ ще вмъкне нов меню елемент `"New Menu Item"` преди „Label 2“.

```
		// here to insert message processing for menu items
		case IDM_ADD_LABEL4:
			hMenu = GetMenu(hWnd);
			hSubMenu = GetSubMenu(hMenu, 3);
			if (GetMenuState(hMenu, IDM_ADD_LABEL4 + 1, MF_BYCOMMAND) == -1)
```

```
{
        ZeroMemory(&miinf, sizeof(miinf));
        miinf.cbSize = sizeof(miinf);
        miinf.fMask = MIIM_ID | MIIM_TYPE | MIIM_STATE;
        miinf.wID = IDM_ADD_LABEL4 + 1;
        miinf.fType = MFT_STRING;
        miinf.dwTypeData = (LPWSTR)_T("New Menu Item");
        miinf.fState = MFS_ENABLED;
        bRes = InsertMenuItem(hMenu, IDM_ADD_LABEL2, FALSE, &miinf );
}
break;
```

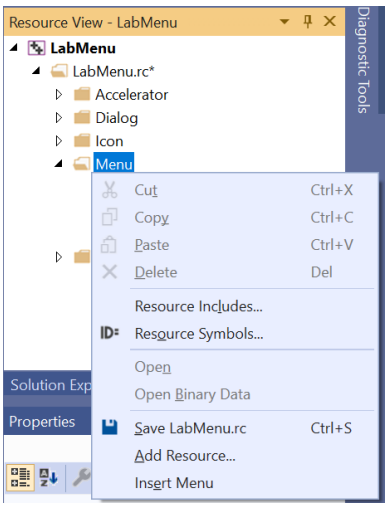
Раздел 2. Работа с контекстно меню

1. Ще създадем контекстно меню (наричано още pop-up меню), което се активира при натискане на десния бутон на мишката в прозореца на приложението, както е представено на следващия изглед:

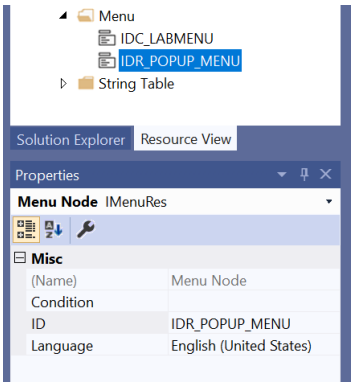


2. Меню елементите “Label 1”, Label 2”, “Label 4” ще изпълняват същите функции като на основния елемент „Add-Delete”, а меню елемент „Message Box“ (IDM_MSGBOX) просто ще изведе подходящ MessageBox. Съобщенията на контекстното меню също се обработват от обслужващата функция на приложението WndProc.
   При контекстно меню акселераторите задавани с “&” спират да работят.

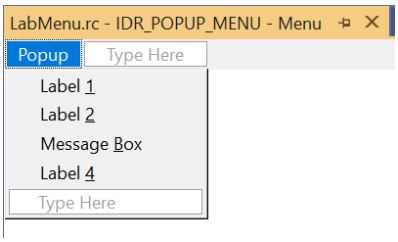
3. Създаваме ново меню. От Resource View с десен клавиш на мишката избираме „Insert Menu”



4. Променяме ID на менюто да стане IDR_POPUP_MENU като го редактираме през Properties прозореца



5. Създаваме изгледа на менюто да изглежда по следния начин:



6. Създаваме обработка на меню елемента „Message Box” по стандартния начин с `case IDM_MSGBOX`:

```
case WM_COMMAND:
    {
        int wmId = LOWORD(wParam);
        // Parse the menu selections:
        switch (wmId)
        {
                // here to insert message processing for menu items
                case IDM_MSGBOX:
                      MessageBox(hWnd, L"Message Box from Pop-up Menu!", L"Info", MB_OK);
                      break;
                case IDM_ADD_LABEL4:
```

7. Остава да предизвикаме показването на контекстното меню на мястото на натискане на десен бутон на мишката. За целта ще добавим обработка на ново съобщение `WM_RBUTTONDOWN`:

```
switch (message)
{
case WM_COMMAND:
    {
        int wmId = LOWORD(wParam);
        // Parse the menu selections:
        switch (wmId) { ... }
    }
    break;
  case WM_RBUTTONDOWN:
        {
                hMenu = LoadMenu(hInst, MAKEINTRESOURCE(IDR_POPUP_MENU));
                hSubMenu = GetSubMenu(hMenu, 0);
                POINT pt = { LOWORD(lParam), HIWORD(lParam) };
                ClientToScreen(hWnd, &pt);
                TrackPopupMenu(hSubMenu, TPM_RIGHTBUTTON, pt.x, pt.y, 0, hWnd, NULL);
                DestroyMenu(hMenu);
                break;
        }
case WM_PAINT:
```