Хедър файлове

Определение

- Хедър файлът е файл, съдържащ декларации, които ще бъдат ползвани в други програмни файлове. Името му съдържа само букви и цифри и завършва с разширение .h или .hpp
- включването на хедър файл в друг става с директивата #include и е равнозначно на копиране на съдържанието на хедър файла на мястото на включване (за разлика от програмните файлове, всеки от които се компилира отделно)

пример:

```
#include <iostream.h>
#include "myheader.hpp"
```

Предназначение

- Системните хедър файлове съдържат декларации, необходими за извикването на функции на системните библиотеки и на операционната система
- Потребителските хедър файлове съдържат декларации на променливи, константи, типове данни и функции, които искаме да са достъпни от други модули
- Дефинициите на функции и всичко, което не искаме да се вижда от другите модули, разполагаме в програмен файл със същото име, в който този хедър файл е включен

Включване с #include

- Системните хедър файлове, разположени в директориите на системните библиотеки, се включват с
  #include <iostream.h>
- Потребителските хедър файлове се разполагат в директорията на проекта или в нейна поддиректория и се включват с
  #include "myheader.h"

Предпазване от повторно включване

За да се предпазим от дублиране на код при повторното включване на хедър файла, той се огражда с директиви за условна компилация:

```
#ifndef HEADER_FILE
#define HEADER_FILE

// съдържание на хедър файла

#endif  // HEADER_FILE
```

Пример за хедър файл

main.cpp:

```
#include "demo.h"

int main(void)
{
        c='*';
        draw(5);
        c='+';
        draw(50);

        return 0;
}
```

demo.h:

```
#ifndef DEMO_HEADER
#define DEMO_HEADER
char c;
void draw(int n);
#endif // DEMO_HEADER
```

demo.cpp:

```
#include "demo.h"
bool nl=false;
void draw(int n)
{
    for (int i=0;i<n;i++)
        cout<<c;
    if (nl) cout<<endl;
}
```

Стъпки при комплилиране на приложение

1. Препроцесорът замества всяко включване на хедър файл чрез копиране на съдържанието на файла на мястото на включването
2. Компилаторът компилира всеки от програмните файлове до обектен файл
3. Свързващият редактор свързва всички обектни файлове (включително и тези на използваните стандартни библиотеки) до изпълним файл, който може да бъде стартиран самостоятелно

Ползи от разделянето на голяма програма на части

- ускоряване на компилацията – прекомпилират се само променените програмни файлове
- по-добра организация на кода – програмният код се разделя на модули и всички функции от даден модул се поставят в един файл
- разделяне на интерфейсът от реализацията – всичко ползвано от други модули се декларира в хедър файла, а останалото се разполага в програмен файл със същото име

Край