І.О. Теплицький, С.О. Семеріков
Криворізький державний
педагогічний університет

Побудова найпростішої системи тестового контрою знань на основі Web-технологій

В курсі інформатики середньої школі Web-технології посідають чільне місце: у навчальній програмі відведено час на вивчення засобів глобальної мережі Інтернет та програмування мовою HTML [1, 2]. Цього цілком достатньо для створення гіпертекстових *навчальних систем* з окремої теми або розділу курсу.

Незважаючи на можливості, що їх надає мова HTML з підвищення активності гіпертексту, за своєю природою він є статичним, а тому – непридатним для побудови *інтерактивних гіпертекстових систем*. Це викликає потребу в об'єднанні можливостей мови HTML з деякою інтерактивною Web-технологією.

На нашу думку, найбільш зручною технологією є CGI (Common Gateway Interface) [3]. Розглянемо основні можливості цієї технології щодо реалізації інтерактивного гіпертекстового контролю знань.

Почнемо з того, що браузер при наборі деякої адреси (URL) з'єднується по протоколу HTTP із вказаним сервером і запитує у нього потрібний файл, приблизно так:

`GET /~cc/cgi-bin/testpage.cgi HTTP/1.0` – це є головне в запиті.

Далі йде інформація, що посилається браузером. Якщо запитується простий HTML-файл, то за умови, що такий файл є, сервер відішле браузеру відповідь:

```
HTTP/1.0 200 Ok
Content-Type: text/html
```

Далі після порожнього рядка (він потрібен, щоб відокремити заголовок від тіла) йде інформація `<HTML><BODY>`...

Якщо файл відсутній, він може бути згенерований програмно за допомогою CGI-сценарію (скрипту). На його базі сервер

сформує і надішле браузеру відповідь, не змінюючи тіла повідомлення, а лише доповнивши заголовок потрібними для протоколу HTTP полями.

Найбільша привабливість CGI – можливість обробки параметрів, що передані скрипту, наприклад, ви можете набрати

`http://www.somehost.edu.ua/somedir/cgi-bin/my_cgi.cgi?param=value`

тобто ви хочете, щоб скрипт `my_cgi.cgi` обробив для вас параметр `param` зі значенням `value`. При запуску CGI-скрипта сервер формує середовище оточення, в якому скрипт може знайти всю доступну інформацію про HTTP-з'єднання і про запит.

Основні змінні оточення:

Змінна	Значення змінної
`REQUEST_METHOD`	Це одне із головних полів, що використовується для визначення методу запиту HTTP. Протокол HTTP використовує методи GET і POST для запиту до сервера. Вони відрізняються тим що, при методі GET запит є частиною URL, тобто `http://www..../myscript.cgi?request`, а при методі POST дані передаються в тілі HTTP-запиту (при GET тіло запиту порожнє). При методі GET запит заноситься в змінну `QUERY_STRING`, а при POST подається на стандартний ввід скрипту.
`QUERY_STRING`	Це рядок запиту при методі GET. Запит з форми кодується браузером, оскільки не всі символи дозволені в URL: всі пропуски замінюються в URL на знак '+', а всі спеціальні і недруковані символи на послідовність %hh, де hh – код символу, роздільник полів форми знак '&', так що при обробці форм треба виконувати декодування.
`CONTENT_LENGTH`	Довжина в байтах тіла запиту. При методі запиту POST необхідно зчитати зі стандартного вводу `CONTENT_LENGTH` байтів, а потім виконати їхню обробку. Звичайно методом POST користуються для передавання форм, що містять потенційно великі області введення тексту. При цьому методі немає ніяких обмежень, а при методі GET існують об-

Змінна	Значення змінної
	меження на довжину URL.
CONTENT_TYPE	Тип тіла запиту (для форм, кодованих вказаним способом, він є application/x-www-form-urlencoded).
REMOTE_ADDR	IP-адреса віддаленого хосту, що виконує даний запит.
REMOTE_HOST	Якщо запитуючий хост має доменне ім'я, то ця змінна містить його, інакше – ту ж саму IP-адресу, що й REMOTE_ADDR.
SCRIPT_NAME	Ім'я скрипту, використане в запиті. Для одержання реального шляху на сервері використовується SCRIPT_FILENAME.
SCRIPT_FILENAME	Ім'я файлу скрипту на сервері.
SERVER_NAME	Ім'я сервера, найчастіше доменне, як www.x.org, але в рідких випадках через відсутність такого може бути IP-адресою, як 157.151.74.254
SERVER_PORT	TCP-порт сервера, що виконує з'єднання. За замовчанням HTTP-порт 80, хоча в деяких випадках може бути іншим.

На рівні протоколу HTTP обмін запитами та відповідями відбувається у 4 етапи:

1. HTTP-клієнт (браузер) з'єднується із сервером за протоколом TCP/IP.
2. Клієнт передає заголовок запиту і, можливо (в залежності від методу), тіло повідомлення запиту. У заголовку обов'язково вказується метод, адреса і версія HTTP.
3. Сервер надсилає відповідь, що складається із заголовка, у якому сервер вказує версію HTTP і код статусу, що може говорити про успішний чи неуспішний результат та його причини. Далі йде тіло відповіді.
4. Розрив TCP/IP з'єднання.

Після рядка запиту йдуть рядки заголовка запиту. Поля заголовка, спільні як для запитів, так і для відповідей:

Поле	Значення поля
Date	Вказує дату запиту

Поле	Значення поля
MIME-version	Вказує версію MIME (за замовчуванням 1.0).
From	Браузер може посилати серверу адресу користувача.
If-Modified-Since	Використовується при методі GET: ресурс повертається, якщо він був змінений з указаного моменту, може використовуватися при кешуванні.
Referer	Містить URL попереднього ресурсу.
User-Agent	Програмне забезпечення клієнта.
Allow	Список методів, підтримуваних ресурсом.
Content-Encoding	Метод кодування, яким закодований ресурс.
Content-Length	Довжина тіла повідомлення.
Content-Type	Містить тип ресурсу (MIME), для текстових ресурсів – кодування символів.
Expires	Дата закінчення дії ресурсу, застосовується для заборони кешування застарілих ресурсів.
Last-Modified	Час останнього оновлення ресурсу.
Accept	Вказує серверу видавати тільки ті формати даних, які клієнт може розпізнати.

Найпростіший запит:

```
GET /index.html HTTP/1.0
```

Більш складний:

```
GET /somedir/somedoc.html HTTP/1.0
User-Agent: Mozilla/2.0
Accept: text/html
Accept: text/plain
Accept: image/gif
```

Передача даних CGI-скрипту через метод GET:

```
GET /~cc/cgi-bin/test.cgi?name=cc&org=kpi&Name=&email=&comment=
HTTP/1.0
User-Agent: Mozila/2.0
Accept: text/html
Accept: image/gif
```

Відповідь HTTP-сервера складається з рядка стану та полів відповіді. Рядок стану має наступний формат:

```
HTTP/version <SP> Status-Code <SP> Status-Phrase
```

де `HTTP/version` – версія протоколу HTTP, `Status-Code` – трьохзна-

чний код, і `Status-Phrase` – текстова фраза, що пояснює код, наприклад: `HTTP/1.0 200 Ok` (200 – код, що означає успішну обробку запиту, та пояснює “Ok”).

У тому випадку, коли запитуваний URL є CGI-скрипт, сервер, базуючись на даних запиту, створює середовище змінних CGI і передає керування скрипту, який повинен видати CGI-заголовок, після якого йде тіло відповіді, генероване скриптом.

Заголовок складається з полів:

Поле	Значення поля
`Content-Type`	Повинне обов’язково бути присутнім, якщо є тіло.
`Location`	Містить URL ресурсу, на який скрипт перенаправляє запит. Як правило, якщо це поле присутнє, більше нічого не вказується.
`Status`	Дозволяє CGI-скрипту повернути статус обробки; якщо це поле не задане, то сервер має на увазі “200 Ok”.

На базі цієї інформації сервер і формує остаточний заголовок, що передається клієнту.

Традиційно в якості програмного забезпечення Web-серверів використовують Apache, який вимагає виділення для нього окремого комп’ютера, що в умовах обмеженої кількості машин в комп’ютерному класі є не найкращим рішенням. Інший варіант – встановлення Apache на кожне робоче місце не завжди є прийнятним з причини його ресурсоємності.

На нашу думку, для створення та використання системи тестового контролю знань на основі технології CGI найбільш доцільним є використання одного з найпростіших вільно поширюваних Web-серверів з відкритим кодом TinyWeb. Для його функціонування необхідна будь-яка операційна система сімейства Windows, а його розмір (менше 50 Кб) та простота використання дозволяють організувати спільну роботу Web-сервера та Web-клієнтів навіть за відсутності мережі в межах одного комп’ютера.

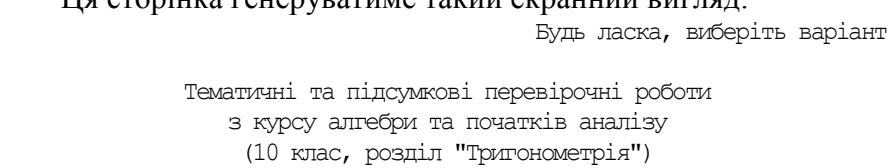
Розглянемо послідовність дій, які необхідно викладачеві виконати для побудови найпростішої тестової системи.

Перш за все, необхідно сформувати початкову сторінку, з

якої учень може перейти до тестування. Нехай ця сторінка має назву main.html. Змістом цієї сторінки повинен бути набір тестів з можливістю вибору варіанту:

```
<HTML><HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-
1251">
<TITLE>Будь ласка, виберіть варіант</TITLE></HEAD>
<BODY BGCOLOR="#ffffff">
<B><P ALIGN="CENTER">Тематичні та підсумкові перевірочні роботи<BR>
з курсу алгебри та початків аналізу<BR>
(10 клас, розділ "Тригонометрія")</P>
</B><P>Тест 1. Тригонометричні формули</P>
<P><A HREF="t1v1/test.html">Варіант 1.</A><BR>
<A HREF="t1v2/test.html">Варіант 2.<BR>
<A HREF="t1v3/test.html">Варіант 3.<BR>
<A HREF="t1v4/test.html">Варіант 4.</P></A>
<P>Тест 2. Тригонометричні функції</P>
<P><A HREF="t2v1/test.html">Варіант 1.<BR>
<A HREF="t2v2/test.html">Варіант 2.<BR>
<A HREF="t2v3/test.html">Варіант 3.<BR>
<A HREF="t2v4/test.html">Варіант 4.</P></A>
<P>Тест 3.Обернені тригонометричні функції </P>
<P><A HREF="t3v1/test.html">Варіант 1.<BR>
<A HREF="t3v2/test.html">Варіант 2.<BR>
<A HREF="t3v3/test.html">Варіант 3.<BR>
<A HREF="t3v4/test.html">Варіант 4.</P></A>
<P>Тест 4. Тригонометричні рівняння</P>
<P><A HREF="t4v1/test.html">Варіант 1.<BR>
<A HREF="t4v2/test.html">Варіант 2.<BR>
<A HREF="t4v3/test.html">Варіант 3.<BR>
<A HREF="t4v4/test.html">Варіант 4.</P></A>
<P>Тест 5. Тригонометричні нерівності</P>
<P><A HREF="t5v1/test.html">Варіант 1.<BR>
<A HREF="t5v2/test.html">Варіант 2.<BR>
<A HREF="t5v3/test.html">Варіант 3.<BR>
<A HREF="t5v4/test.html">Варіант 4.</P></A>
</BODY></HTML>
```

Ця сторінка генеруватиме такий екранний вигляд:



```
Тест 1. Тригонометричні формули

Варіант 1.
Варіант 2.
Варіант 3.
Варіант 4.

Тест 2. Тригонометричні функції

Варіант 1.
Варіант 2.
                                <...>
```

Нехай учнем обрано третій варіант першого тесту. Перехід до нього виконується за посиланням

```
<A HREF="t1v3/test.html">Варіант 3.<BR>
```

Всі файли з назвою `test.html` мають таку структуру:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML><HEAD><TITLE>Інструкція з тестування</TITLE></HEAD>
<BODY BGCOLOR=#FFFFFF>
<FORM METHOD=GET ACTION="http://localhost/cgi-bin/copyt1v3.exe">
  <TABLE><TR>
      <TD VALIGN=TOP>Вам пропонується пройти тестування з курсу алгебри та початків аналізу середньої школи. На Вас чекають ряд запитань, для відповіді на які необхідно обрати один з трьох варіантів:</TD>
      <TD>
        <INPUT TYPE=RADIO NAME="rad" VALUE="t000on1" CHECKED>№1<BR>
        <INPUT TYPE=RADIO NAME="rad" VALUE="t000on2">№2<BR>
        <INPUT TYPE=RADIO NAME="rad" VALUE="t000on3">№3<BR>
      </TD>
  </TR></TABLE>
<BR><INPUT TYPE=submit VALUE="Розпочати тестування"> 
</FORM>
</BODY></HTML>
```

Змінною частиною у цьому документі є посилання на скрипт:

```
<FORM METHOD=GET ACTION="http://localhost/cgi-bin/copyt1v3.exe">
```

Ім’я скрипту формується з номеру тесту та варіанту у ньому. Єдина дія, що виконує ця програма – копіювання даних, необхідних для початку тестування та виклик тестового скрипту:

```
main() { system("copy ..\\t1v3\\* ."); system("controls.exe"); }
```

Зрозуміло, що цю програму можна з легкістю замінити відповідним командним файлом. Єдина причина, за якою це не зроблено – відсутність підтримки bat-файлів сервером TinyWeb у якості CGI-скриптів.

Екранне подання наведеного HTML-тексту може бути таким:

```
                                                Інструкція з тестування

   Вам пропонується пройти тестування з курсу алгебри та початків
аналізу середньої школи. На Вас чекають ряд запитань, для відповіді на
які необхідно обрати один з трьох варіантів:
                                    (*) №1
                                    ( ) №2
                                    ( ) №3
   Розпочати тестування
```

За замовчанням обраною є перша кнопка – це досягається командою

```
<INPUT TYPE=RADIO NAME="rad" VALUE="t000on1" CHECKED>№1<BR>
```

В процесі тестування кнопка виступає у якості поля введення, значення від якого аналізує скрипт `controls.exe`. Для початку тестування необхідно обрати будь-яку кнопку та перейти до поля, яке викликає скрипт:

```
<INPUT TYPE=submit VALUE="Розпочати тестування">
```

Відповідь, що надсилається скриптом, має такий вигляд:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML><HEAD><TITLE>Тестування</TITLE></HEAD>
<BODY BGCOLOR=#FFFFFF><H2>Завдання №1</H2>
<FORM METHOD=GET ACTION="http://localhost/cgi-bin/controls.exe">
<P>Виразити 12<sup>o</sup> у радіанах:<P>
<INPUT TYPE=RADIO NAME="rad" VALUE="t001on1" CHECKED>pi/15<P>
<INPUT TYPE=RADIO NAME="rad" VALUE="t001on2">pi/45<P>
<INPUT TYPE=RADIO NAME="rad" VALUE="t001on3">pi/30<P><BR>
<INPUT TYPE=submit VALUE="Відповісти"> </FORM>
</BODY></HTML>
```

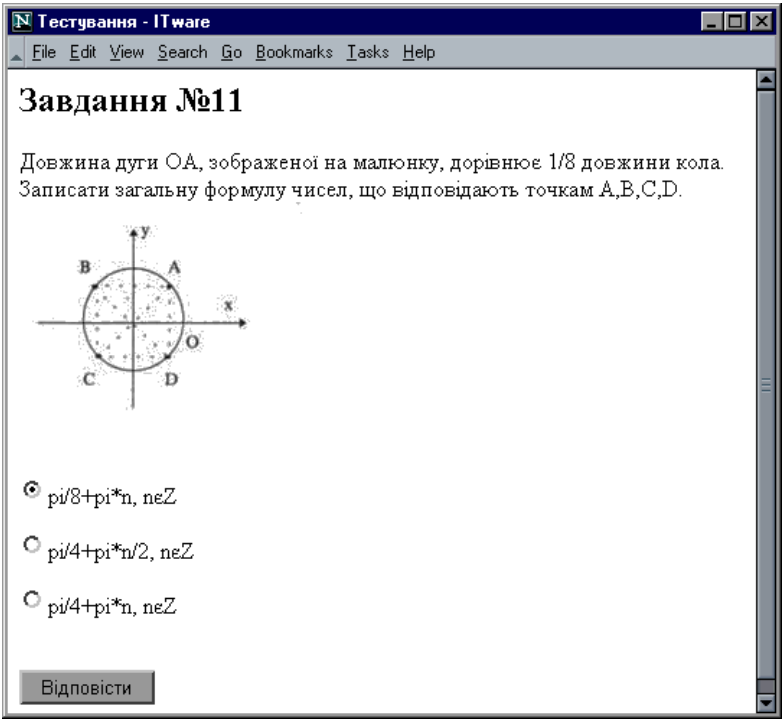
Дані для цього завдання беруться з файлу `tests.txt`, що має таку структуру:

```
15
Виразити 12<sup>o</sup> у радіанах:
pi/15
pi/45
```

```
pi/30
1
pi/9=...
5<sup>o</sup>
10<sup>o</sup>
20<sup>o</sup>
3
...
```

Перший рядок файлу – кількість тестових завдань, що міститься в ньому. На кожний тест необхідно задати 5 рядків: 1 – умова завдання, 2 – перший варіант відповіді, 3 – другий варіант відповіді, 4 – третій варіант відповіді, 5 – номер правильної відповіді.

Як у завданнях, так і у відповідях можуть використовуватися будь-які теги гіпертекстової мови, що транслюються в екранне відображення безпосередньо браузером. Це дозволяє використовувати в тесті малюнки, відео та звукові фрагменти тощо:



Оцінювання знань відбувається за *умовною* дванадцятибальною шкалою через знаходження відношення кількості правильних відповідей до загальної кількості завдань:

```
                                                    Тестування

Тестування завершено

   Ви правильно відповіли на 4 питань з 15

   Оцінка - 3

   Повернутися до вибору варіанту
```

Головна частина системи – це тестуючий скрипт, що має такий зміст:

Код	Коментар

`#include <stdio.h>`
`#include <stdlib.h>`
`#include <string.h>`
`#include <vector>`
`#include <cstring.h>`
`#include <fstream.h>`
	Заголовочні файли мови С++, що необхідні для роботи програми

`struct q` `{`	Означення типу “питання”, що відповідає за зберігання даних про питання
`  std::string question,` `  var1, var2, var3;`	Рядки питання та варіанти відповідей
`  int answer;`	Номер правильної відповіді
`  q():question(""),var1(""),var2(""),` `var3(""),answer(0){}`	При створенні змінної типу спочатку всі поля порожні

`  q operator=(const q &x) {`
`    question=x.question;`
`    var1=x.var1;`
`    var2=x.var2;`
`    var3=x.var3;`
`    answer=x.answer;`
`    return *this;`
`  }`
	Оператор копіювання – необхідний для зберігання типу “питання” у векторному класі

`  bool operator==(const q &x) {`
`    return question==x.question&&`
`  var1==x.var1&&var2==x.var2&&`
`  var3==x.var3&&`
`  answer==x.answer;`
`  }`
	Оператор порівняння – необхідний для зберігання типу “питання” у векторному класі

Код	Коментар
`};`	Створюємо порожній вектор елементів типу “питання”
`std::vector<q> tests;`	
`void main() {`	Точка входу у скрипт
`  FILE * fp;`	Файлова змінна
`  char * query;`	Рядок запиту
`  char buf[301];`	Буфер для зберігання одного рядка
`  int count,`	Кількість тестів та номер поточного тесту
`      curtest;`	
`  q temp;`	Змінна типу “питання”
`  cout<<"Content-type: "`	Виведення заголовка HTTP та порожнього рядка після нього
`        "text/html\n\n";`	
`  cout<<"<!DOCTYPE HTML PUBLIC"`	Виведення початкового фрагмента документа HTML
`    " \"-//W3C//DTD HTML 3.2//EN\">";`	
`  cout<<"<HTML><HEAD><TITLE>"`	Виведення заголовку сторінки та задання її кольору
`        "Тестування</TITLE></HEAD>"`	
`        "<BODY BGCOLOR=#FFFFFF>";`	
`  ifstream f("tests.txt");`	Відкриття файлу з тестами
`  f.getline(buf,300);`	Зчитування першого рядку файлу – кількості тестів
`  count=atoi(buf);`	
`  for(int i=0;i<count;i++) {`	
`     f.getline(buf,300);`	
`     temp.question=std::string(buf);`	
`     f.getline(buf,300);`	
`     temp.var1=std::string(buf);`	До тих пір, поки не прочитаємо усі тести, зчитуємо питання, варіанти та номер відповіді, заносячи їх у змінну типу “питання”, та поповнюємо вектор цим питанням
`     f.getline(buf,300);`	
`     temp.var2=std::string(buf);`	
`     f.getline(buf,300);`	
`     temp.var3=std::string(buf);`	
`     f.getline(buf,300);`	
`     temp.answer=atoi(buf);`	
`     tests.push_back(temp);`	
`  }`	
`  query = getenv("QUERY_STRING");`	Отримуємо дані від форми. При використанні методу GET вони передаються через змінну середовища QUERY_STRING
`  if(!strncmp(query+1+4,"000",3))`	Номер тесту 0 відповідає початку тестування
`  {`	

Код	Коментар
`curtest=0;`	Встановлюємо номер поточного тесту в 0
`fp=fopen("received.dat", "w+b");`	Відкриваємо файл, в якому зберігатимуться проміжні результати тестування
`}`	
`else {`	Якщо тестування вже почалося
`curtest=(query[1+4]-'0')*100+ (query[2+4]-'0')*10+ (query[3+4]-'0');`	Одержуємо зі змінної оточення номер поточного тесту
`fp=fopen("received.dat", "r+b");`	Відкриваємо файл, в якому зберігатимуться проміжні результати тестування
`fseek(fp,curtest-1,SEEK_SET);`	Виконуємо зміщення у файлі в позицію, що відповідає попередньому тесту
`fwrite(query+6+4, 1, 1, fp);`	Заносимо у файл вибір, зроблений у відповідь на попередній тест
`}`	
`fclose(fp);`	Закриваємо файл
`if(curtest!=count) {`	Якщо тестування ще не закінчилося, надсилаємо у відповідь наступне завдання
`cout<<"<H2>Завдання №" <<curtest+1<<"</H2>";`	Виводимо номер завдання
`cout<<"<FORM METHOD=GET ACTION=\"http://localhost/cgi-bin/controls.exe\">";`	Виводимо вказівку про те, що сторінка містить посилання на скрипт
`cout<<"<P>" <<tests[curtest].question;`	
`cout<<"<P>";`	
`printf("<INPUT TYPE=RADIO NAME" "=\"rad\" VALUE=\"t%03don1" "\" CHECKED>%s",curtest+1, tests[curtest].var1.c_str());`	Формуємо зміст сторінки: виводимо питання та варіанти відповідей, пов'язані з кнопками
`cout<<"<P>";`	
`printf("<INPUT TYPE=RADIO NAME" "=\"rad\" VALUE=\"t%03don2" "\" CHECKED>%s",curtest+1, tests[curtest].var2.c_str());`	
`cout<<"<P>";`	

Код	Коментар
printf("<INPUT TYPE=RADIO NAME" "=\"rad\" VALUE=\"t%03don3" "\" CHECKED>%s",curtest+1, tests[curtest].var3.c_str());	
cout<<"<P>";	
cout<<" <INPUT TYPE=submit VALUE=\"Відповісти\"> "; }	Виводимо посилання, що викликатиме вказаний скрипт
else {	
cout<<"<H2>Тестування " "завершено</H2>";	Якщо тестування завершено, виводимо відповідне повідомлення
fp=fopen("received.dat", "r+b"); int right=0;	
for(int i=0;i<count;i++) { fread(buf, 1, 1, fp); if(buf[0]-'0'==tests[i].answer) right++; }	Відкриваємо файл з відповідями та підраховуємо кількість правильних відповідей
fclose(fp);	Закриваємо файл
cout<<"Ви правильно відповіли на "<<right<<" питань з "<<count;	Виводимо кількість правильних відповідей та оцінку
cout<<"<P>Оцінка - " <<12*right/count;	
//cout<<"<P><A HREF=\"test.html\" TARGET=\"main page\">Повторити тестування</P>";	
cout<<"<P><A HREF=\"../main.htm\" TARGET=\"main page\">Повернутися до вибору варіанту</P>";	За бажанням – повторюємо тестування чи повертаємося до вибору варіанту
}	
cout<<"</FORM></BODY></HTML>"; }	Виводимо кінець форми та документу

Апробація показала, що цього скрипту цілком вистачає для побудови простої однокористувацької тестової системи з вибором відповіді, для наповнення якої достатньо мати найпростіші текстовий та графічний редактори.

Модифікація пропонованого скрипту дозволяє розширити номенклатуру тестів асоціативними, багатовибірковими, з можливістю довільної текстової відповіді тощо.

Література:

1. Морзе Н.В., Козачук О.В., Жалдак М.І. Вивчення основ комп'ютерних мереж // Комп'ютер у школі та сім'ї. – 2000. – №2. – С. 14–18.
2. Рамський Ю.С., Іваськів І.С. Методика навчання основ Web-програмування в загальноосвітній школі // Комп'ютер у школі та сім'ї. – 2000. – №1–4.
3. Фролов А.В., Фролов Г.В. Сервер Web своими руками. – М.: Диалог-МИФИ, 1997. – 288 с. (Библиотека системного программиста; Т. 29).