Тема: Основи роботи в середовищі Borland C++.

Мета: ознайомитися з середовищем мови програмування Borland C++ Builder, реалізувати програму для найпростіших дій введення та виведення інформації.

Теоретичні відомості

Основні функції роботи у консольному режимі

Консольний додаток - це програма, для якої пристроєм введення інформації є клавіатура, а пристроєм виведення - монітор у символьному режимі.

Консольний режим - це режим створювання програм користувача без використання графічного інтерфейсу, а з відображенням лише символьної інформації. В операційній системі робота у консольному режимі здійснюється засобами командного рядка.

Виконування консольного додатка відбувається у чорному консольному екрані. Для введення даних слід використовувати спеціальні команди. Під час виконування вони скомандують програмі зупинитися й очікувати, допоки користувач уведе значення і натисне клавішу <Enter>. Виведення даних виконується у теж саме чорне вікно і завжди потребує вказівок та пояснень для користувача.

Виведення означає, що певний текст чи то значення певних змінних висвітяться на екрані і користувач зможе їх прочитати. Введення означає, що користувач задає значення змінної, яке буде використано у програмі.

Створення консольного додатка

Перед тим як розпочати створювання консольного програмного додатка, слід розглянути функції, які забезпечують введення даних з клавіатури та виведення результатів на екран.

Для виведення тексту у C++ найчастіше використовується команда cout<< і текст записується у подвійних лапках, наприклад:

```
cout << «Текст»;
```

Для виведення значення змінної також використовується команда cout<< із зазначенням імені змінної без лапок. Наприклад, щоб вивести значення змінної x, слід написати:

```
cout << x;
```

Для виведення значення змінної x з коментарем, слід написати:

```
cout << «Значення змінної x: » << x;
```

Після кожного виведення курсор залишається на тому ж самому рядку і наступне виведення виконується поряд з попереднім. Якщо треба перемістити курсор на новий рядок після виведення даних, слід написати <<endl наприкінці команди:

```
cout << « Значення змінної x: » << x << endl;
```

Для введення значення змінної зазвичай використовується команда

cin>>:

```
cin >> x;
```

Після того як програма завершила свою роботу, чорне вікно закривається. Зазвичай це трапляється настільки швидко, що користувач не встигає побачити результати виконання програми. Щоб зупинити програму і надати користувачу можливість прочитати результати, використовується функція getch(), яка очікує на натиснення будь-якої клавіші, наприкінці програми перед return 0: getch();

Аналогічну дію виконує функція cin.get ().

Зверніть увагу! Щоб у програмі можна було використовувати команди cin та cout, на початку програми слід написати:

```
#include <iostream.h>
```

Для використання функції getch() слід на початку програми написати:

```
#include <conio.h>
```

Окрім специфічних команд C++: cin та cout (вони мають назву команд *потокового введення-виведення),* - існують команди для введення-виведення, які прийшли до C++ з C: scanf та printf. Ці команди мають назву *команд форматного введення-виведення.*

Для того щоб програма могла використовувати згадані команди, на початку програми слід долучити директиву #include <stdio.h> .

У загальному вигляді функція scanf для введення значення однієї змінної виглядає як

scanf*(<формат>, &<змінна>);*

де: *формат* - рядок специфікаторів формату у подвійних лапках. Найбільш поширеними специфікаторами є: %i - для введення цілих чисел, %f - для введення дійсних чисел, %s - для введення рядка символів;

змінна - це ім’я змінної, значення якої вводиться.

Знак & є елементом синтаксису і означає операцію отримання адреси змінної у пам’яті.

Послідовність створювання консольного програмного додатка

Щоб створити в C++ Builder консольний програмний додаток, слід виконати такі дії. Спочатку в меню File слід обрати команду New... Other і на багаторядковій панелі New Items діалогового вікна (рис.) клацнути по значкові Console Wizard.

Внаслідок виконаних дій на екрані з’явиться вікно Console Wizard (рис.). В цьому вікні можна обрати мову програмування і зазначити, чи буде використовуватись та чи інша бібліотека. Після того як буде задано параметри створюваного консольного додатка, слід клацнути по кнопці ОК.

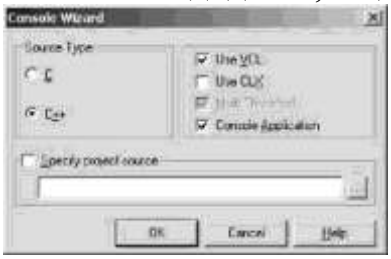


Рис. Вікно Console Wizard

Як наслідок C++ Builder створить проект консольного додатка - і на екрані з’явиться вікно редактора коду Unitl.cpp (рис.), в якому буде створено шаблон консольного додатка чи то функція main().

Рис. Вікно редактора коду Unit1.cpp

Розпочинається консольний додаток директивою #pragma hdrstop, яка забороняє виконання попередньої компіляції долучених файлів.

Після цієї директиви можна вставити директиви #include, які забезпечують долучення потрібних заголовних файлів. Наприклад #include <stdio.h> долучає заголовний файл, який містить прототипи функцій введення-виведення, у тому числі printf(). Файл conio.h потрібен, оскільки для очікування натиснення клавіші було застосовано функцію введення символу getch().

Директива #pragma arqsused скасовує попередження компіляції про те, що аргументи, зазначені у заголовку функції, не використовуються.

Функція main () є присутня у кожній програмі, саме через неї передається керування після завантаження та ініціалізації програми.

Слід звернути увагу на те, що консольний додаток розробляється у Windows, а виконується як програма DOS. Оскільки в DOS та Windows літери кирилиці мають різні коди, це призводить до того, що консольний додаток замість повідомлень на кирилиці виводить “абракадабру”.

Для зберігання проекту слід обрати у головному меню File/Save Project as... і зберегти файли проекту в окремому каталозі (теці).

Для компіляції та запуску програми на виконання треба натиснути кнопку - Run на інструментальній панелі чи функціональну клавішу <F9>, після чого на екрані з’явиться прототип вікна MS DOS.

Якщо натиснути будь-яку клавішу на клавіатурі, - програма завершиться і її вікно закриється.

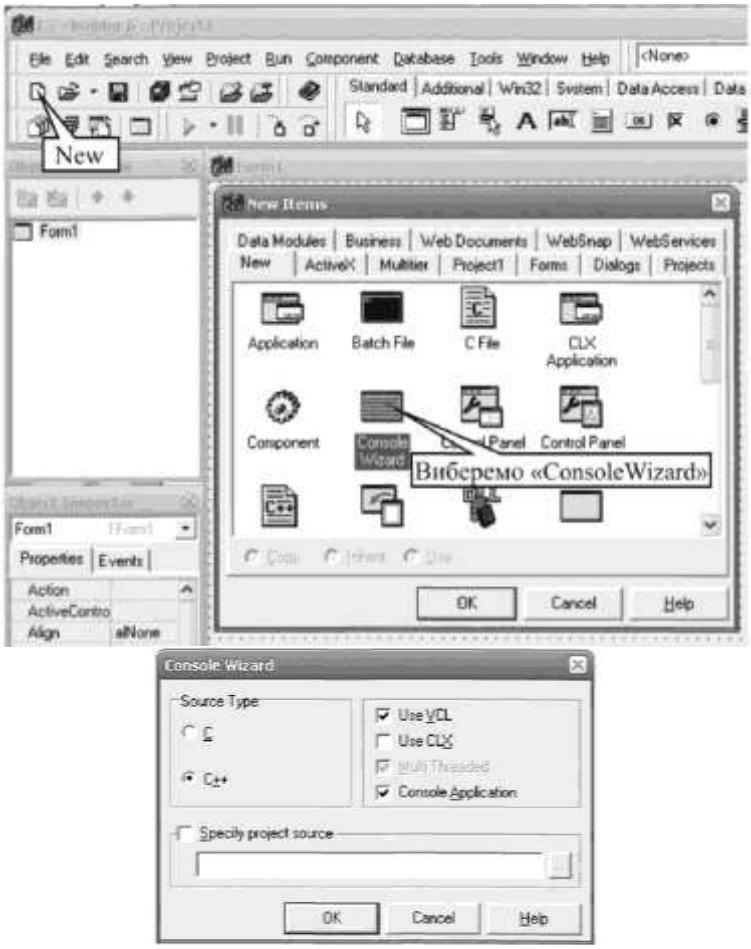
ХІД РОБОТИ

Завдання 1.

Перша програма

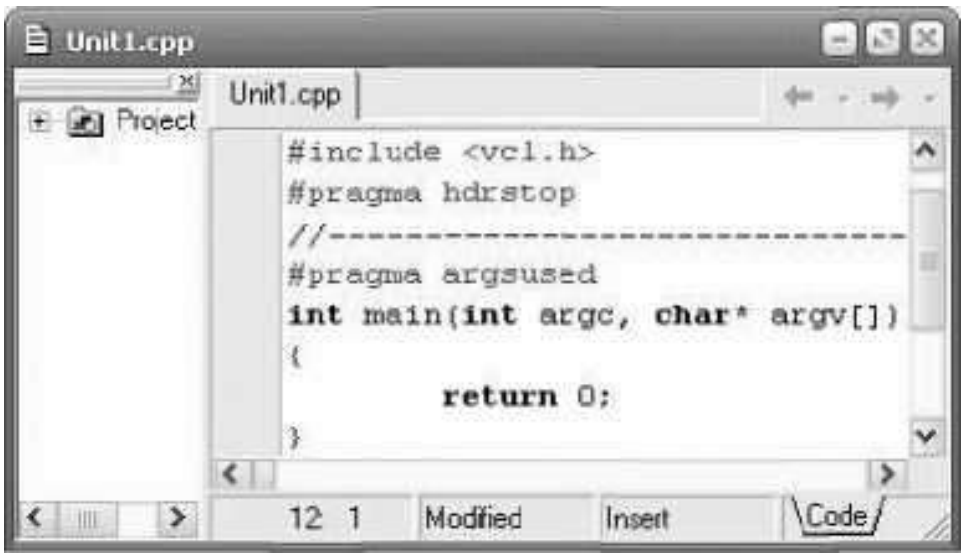
Для запуску програми натисніть: «Пуск» —> «Програми» —> «Borland C++Builder6» —> «C++Builder6».

Для введення вашої програми виберіть на панелі кнопку «New» (див. мал.), що означає створення нового проекту. Відкриється вікно «New items». Виберемо «ConsoleWizard».



У діалоговому вікні «Console-Wizard» виберемо мову програмування C++ та натиснемо «ОК». В результаті C++ Builder створить проект консольної програми, тобто такої, що виконуватиметься без використання графічного інтерфейсу Windows. На екрані з'явиться вікно редактора коду, в якому знаходиться шаблон програми - функція main, детальніше про яку ви дізнаєтесь пізніше. Це вікно далі можна використовувати для уведення та редагування програми.

Уведіть вашу першу програму, яка є лінійною програмою. Для цього допишіть виділені жирним шрифтом рядки, не заглиблюючись поки що в їхній зміст:



//Програма 1.1. *Студент Іванов І.І., група 31 Комп*

```
#include <vcl.h>
#include<iostream.h>
#include<conio.h>
#pragma hdrstop
#pragma argsused
int main()
{
cout << "My first program!";
getch();
return 0;
}
```

Для того щоб програма була виконана і видала результат, необхідно перетворити її у послідовність команд процесора. Цю функцію виконує компілятор - програма-перекладач з мови програмування (в нашому випадку С++) на мову машинних кодів, зрозумілих процесору. Щоб відкомпілювати вашу програму і запустити на виконання, виберіть пункт меню «Run» чи натисніть клавішу F9. Якщо ви не припустилися синтаксичних помилок, про що вкаже компілятор у спеціальному вікні, результат роботи програми буде таким:

My first program!

Поекспериментуйте із цією програмою, щоразу компілюючи і запускаючи після внесення змін до тексту:

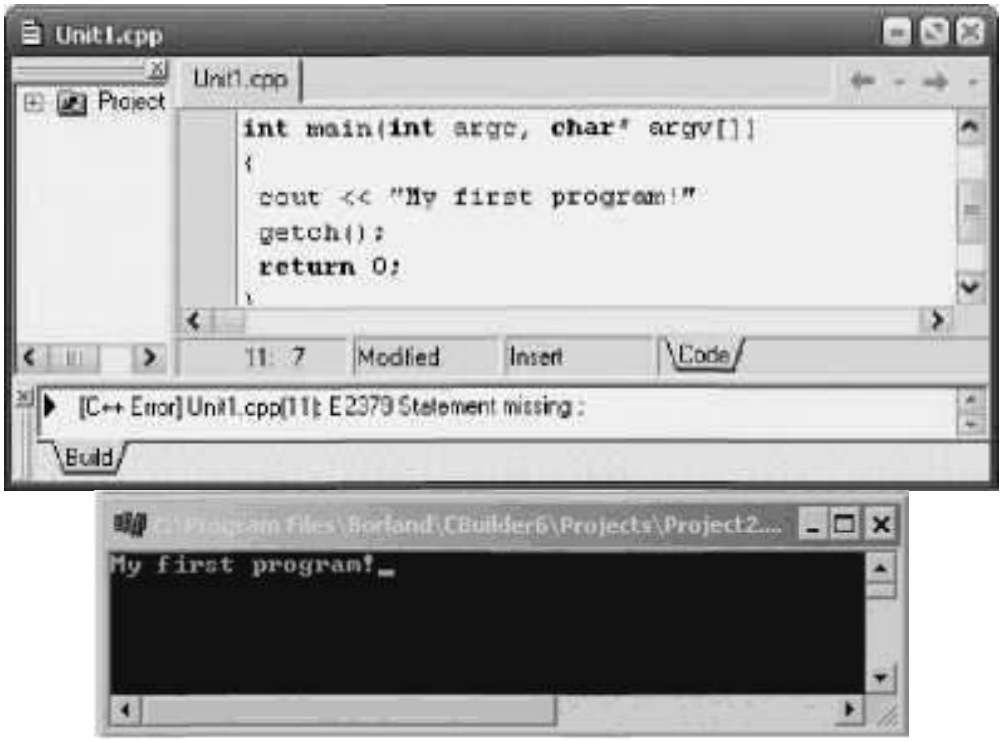
```
cout << "My first program!" << endl;
```

Потім так:

```
cout << "Programuvati prosto!
cout << "Duge prosto!" << endl;
```

Ім'ям endl позначено спеціальний символ - кінець рядка (англ, end *line* - кінець рядка: читається «енд-ел»). При виведенні він спричиняє перехід на новий рядок.

Програмуючи у Borland C++Builder6 уводьте текст в лапках латинськими буквами замість українських. Це пов'язано з тим. що в DOS і Windows для російських та українських літер використовують різні кодування (DOS - ASCII, а Windows - ANSI). Це призводить до того, що при виконанні програм замість очікуваного тексту на екран виводиться якась нісенітниця. Для коректного виведення літер кирилиці ми потім напишемо спеціальну функцію.



Спробуйте не брати текст у рядку з cout у лапки - компілятор виявить синтаксичну помилку, про що буде вказано у рядку з відповідним повідомленням (див. мал.). Клацніть по ньому мишкою й компілятор вкаже вам на рядок з помилкою.

На малюнку видно, що програміст забув поставити крапку з комою в кінці оператора. Компілятор знайшов помилку і вивів повідомлення: «Statement missing;», вказавши також ім'я файлу та номер рядка з помилкою (Unitl. cpp (11)).

При створенні програми мовою C++ ви повинні дотримуватися певних правил. Наприклад, треба брати текстові повідомлення в лапки й ставити крапку з комою в кінці більшості операторів C++. Тільки виправивши всі помилки ви одержите результат виконання програми.

Оператори можна записувати в один рядок, але краще, коли кожен оператор буде починатися з нового рядка. Така програма буде зрозуміліша і в ній легше буде знайти помилки.

Розглянемо докладніше вашу першу програму. Вона складається із

директив препроцесора, декількох операторів, функції і фігурних дужок, які називають групуючими символами:

```
#include <iostream.h> //директиви препроцесора
#include <conio.h>
int main()                    //заголовок функції
{
cout<<"My first program" //продовження оператора
<<"C++!";    //у новому рядку не є помилкою
getch();
return 0;
}
```

Для кращого розуміння програм, у них корисно вставляти коментарі, перед якими ставляться дві косі риски («подвійний слеш» //) (див. текст програми). Компілятор ніяк не обробляє інформацію, розміщену після цих символів до кінця рядка. Якщо коментар займає кілька рядків, то на його початку ставлять символи /*, а в кінці - */.

Заголовкові Файли (бібліотеки)

Препроцесором називається перша фаза компілятора. Інструкції препроцесора називаються **директивами** (розпорядженнями). Рядки, що їх містять, починаються зі знака #. Практично кожна програма на C++ містить директиви препроцесора #**include** (читати «паунд інклуд»). Вони необхідні для включення у програму певних файлів під час компіляції.

Наприклад, для виведення на екран тексту за допомогою об'єкта **cout** (читати «сі-аут») та операції перенаправления вихідного потоку << (будемо далі називати операцією виведення), необхідно підключити файл *<iostream.h>* (читати «ай-оустрім крапка ейч» від англійських: input-output streams - вхідні-вихідні потоки). Для застосування різних математичних функцій слід підключити файл *<math.h>;* для затримки зображення на екрані використовуємо функцію **getch** (), яка повертає код символу натиснутої клавіші, а для її роботи підключаємо бібліотеку ***<conio.h>*** і т.д.

Файли з розширенням *h* називаються **заголовковими.** Розташовуються заголовкові файли в підкаталозі *INCLUDE,* і ви можете переглянути вміст цих файлів, але змінювати їх забороняється!

Головна функція

Кожна програма на C++ має один вхід, з якого починається виконання програми, - головну функцію, до складу якої входить перший виконуваний оператор.

Запис

int main (int argc, char* argv[]), чи

int main ().

визначає головну функцію. Ваші програми повинні завжди включати одну і

тільки одну функцію з ім’ям main.

Слово int перед назвою функції означає, що дана функція повертає значення типу int. В дужках після main у першому прикладі перелічені аргументи командного рядка. Такий варіант запису (з аргументами командного рядка (int argc, char* argv[])) слід використовувати, якщо передбачається обробка програмою вхідних аргументів. Зараз на аргументи командного рядка ми уваги не звертатимемо. Можна просто вилучити їх опис, якщо система згенерувала його автоматично, і використовувати другий варіант заголовка функції.

Крім головної функції програма може містити інші функції.

За правилами C++, до функції main висуваються особливі вимоги. Зокрема, її слід оголошувати такою, що повертає значення типу int. Повернення нею значення 0 (оператор return 0;) означає, що виконання функції main, а отже й всієї програми, завершилось успішно. Інші значення є кодами повідомлень про помилки, які трапились в ході виконання програми.

Групуючі символи ***{}***

Служать для групування зв'язаних операторів. У простих програмах, які ми надалі будемо складати, ці символи будуть групувати оператори, які відповідають операторам вашої головної функції, чи допоміжної, якщо така буде використовуватися. Пізніше ви вивчите інші випадки застосування групуючих символів.

Виведення повідомлень на екран

Як було сказано вище, для виведення даних на екран використовується об’єкт cout та операція виведення <<.

Розглянемо приклад виведення чисел на екран:

```
//Програма 1.2. Студент Іванов І.І., група 31 Комп
#include<iostream.h>
#include<conio.h>
int main()
{
cout << 1001;
getch(); return 0;
}
```

Спробуйте вивести дійсне число (їх часто називають числами із плаваючою крапкою):

```
cout<<0.8976;
```

Потім направте у вихідний потік числа в такий спосіб:

```
cout<<l<<2<<0<<0<<l;
```

Перевірте, як спрацюють такі рядки:

```
cout<<"Vvedi ocinku: "<<12<<endl;
```

cout<<"Ocinka "<<12<<" - uljublena."<<endl;

Отже, у вихідний потік можна виводити як числа, так і текстові повідомлення. Текст при цьому слід брати в лапки. В одному операторі можна вивести послідовно декілька числових та текстових значень.

Спеціальні символи виведення

Випробуйте таку програму:

```
//Програма 1.3. Студент Іванов І.І., група 31 Комп
#include<iostream.h>
#include<conio.h>
int main()
{
cout<<"Ryad l\nRyad 2";
getch();
return 0;
}
```

А тепер змініть програму так:

```
cout<<1<<'\n'<< 0<<'\n'<<3;
```

Послідовності символів, що починаються **із** символу «\» («зворотний слеш») називають **керуючими послідовностями** або **спеціальними символами.** Вони зображаються на екрані двома символами, але компілятором сприймаються як один символ.

У наведених прикладах використана керуюча послідовність **\n** — символ нового рядка, яка при виведенні поміщає курсор у початок наступного рядка, аналогічно до **endl** (кінець рядка).

Призначення спеціальних символів

Символ	Призначення
\a	Звуковий сигнал (дзвінок)
\b	Крок назад (зворотний пропуск)
\f	Перехід на нову сторінку
\n	Перехід на новий рядок
\r	Повернення каретки (не перехід на новий рядок!)
\t	Символ горизонтальної табуляції
\v	Символ вертикальної табуляції
****	Символ «зворотний слеш»
\?	Знак питання
\'	Одинарні лапки
\"	Подвійні лапки
\0	Нульовий символ

Символ «зворотний слеш» використовується для включення в рядок:

- кодів, що не мають графічного зображення (наприклад, **\a -** звуковий сигнал, **\n -** переведення курсору в початок наступного рядка);
- символів апострофа (’), зворотного слеша (\), знака питання (?) і лапок (").

Спеціальні символи розташовуються або у одинарних лапках, якщо використовуються окремо, або у подвійних, разом з іншими символами рядка.

Наприклад, якщо всередині рядка потрібно записати лапки, їх випереджають косою рискою, за якою компілятор відрізняє їх від лапок, що обмежують рядок:

"Видавництво\"Аспект\" "

Коли ми вивчатимемо рядки, то побачимо, що у кінець рядкової константи компілятор додає спеціальний символ NULL (тобто ‘\0’).

Спробуйте дослідити цю програму:

```
//Програма 1.4
#include<iostream.h>
#include<conio.h>
int main()
{
cout<<"Dzvon! \a\t Dzvon!\a\t"; getch();return 0;
}
```

<u>Ширина виведення</u>

Модифікатори формату використовуються для керування шириною поля, що відводиться для розміщення значення, яке виводиться. Модифікатор setw дозволить вам регулювати кількість символів, займаних виведеним числом. Але при цьому ви повинні включити в програму заголовковий файл *<iomanip.h>*:

```
//Програма 1.5
#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
int main()
{
cout<<"Druk:"<<setw(3)<<1012<<endl;
cout<<"Druk:"<<setw(4)<<1012<<endl;
cout<<"Druk:"<<setw(5)<<1012<<endl;
cout<<"Druk:"<<setw(6)<<1012<<endl;
getch();
return 0;
}
```

При використанні **setw** *ви вказуєте* ***мінімальну*** *кількість символьних позицій, займаних числом.*

У програмі 1.5 модифікатор **setw**(3**)** вказує мінімум 3 символи, однак у зв'язку з тим, що число 1012 потребує більше трьох символів, об'єкт **cout** з операцією виведення << використовує реально необхідну кількість символів. Для кожного виведеного значення використовується окремий модифікатор **setw.**

Збереження програм у Borland C++Builder6

Для збереження вашої програми (проекту) виконайте такі дії: «File» —» «Save Project As...». У вікні «Save Unitl As» виберіть папку, наприклад: *C:\Студентам\Іванов* та натисніть «Создание новой папки». Назвіть її *pr1*. Потім відкрийте папку *pr1* і для першого завдання знову створіть папку, наприклад *1*. Відкрийте її і натисніть «Сохранить» для файлу з ім'ям *Unitl.cpp* а також для файлу *Projectl.bpr*.

Зверніть увагу: програма, яку ви випробовували збережена у файлі *Unitl.cpp,* а файл *Projectl.bpr* з описом загальних властивостей проекту був створений автоматично.

Щоб відкрити проект, виберіть «File» —» «Open Project» і у діалоговому вікні, що з'явиться, виберіть потрібну папку та відкрийте файл *Projectl.bpr*.

Завдання 2.

Виконати та випробувати усі програми 1.1 - 1.5 з усіма вказаними доповненнями.

Завдання 3.

Збережіть програми, створивши у власній папці нову папку *Pr1*.

Примітка. Якщо власна папка ще не створена, її назву та розміщення слід уточнити у викладача. Повний шлях, орієнтовно, може відповідати такому шаблону:

<диск>:\<Студентам>\<прізвище>\<вправа>.

Завдання 4

Дайте в звіті відповіді на контрольні запитання.

Завдання 5

Зробіть висновок по роботі.

Контрольні запитання:

1. З чого складається алфавіт мови С++?
2. Які службові слова використовує мова С++?
3. Що являє собою структура програми? Наведіть основні вимоги, які слід ураховувати при створенні програм мовою С++.
4. Що таке консольний додаток?
5. Які файли називаються заголовковими?
6. Що таке «головна функція»?
7. За допомогою якої директиви препроцесора до програми долучають бібліотечні модулі (заголовні файли)?
8. В яких бібліотеках містяться математичні функції?
9. Яке призначення мають фігурні та круглі дужки в С++?
10. Назвіть послідовність створювання програмного проекту у С++ Builder.
11. В який спосіб можна зберегти програмний проект?
12. В які способи можна запустити проект на виконання?
13. Які функції введення-виведення даних у консольному режимі Вам відомі?
14. Які заголовні файли слід долучити для використання функцій введення-виведення?
15. Визначите, що являють собою наступні керуючі послідовності: а) \n; б) \\; в) \” д) \t
16. Які пріоритети виконання операцій.
17. Як об’явити змінну.
18. Що таке «компілятор»?
19. Як компілятор повідомляє про помилку у програмі?
20. Назвіть спеціальні символи виведення. Для чого вони призначені?