Лекція (char)(~0>>5)

Робота з C++. Потоки введення/виведення. Посилання. Оператори new/delete

План

1. Прості програми на Сі++
2. Потокове ведення/виведення на Сі++
- Оператор ::
- Введення/виведення cin/cout
- Форматування виводу
- Маніпулятори
- Робота з текстовими файлами
3. Булевий тип
4. Перевантаження функцій
5. Посилання та використання посилань
6. Виділення динамічної пам’яті через new та delete

C-style код в C++

Рецепт: додати ‘c’ до назви бібліотеки, видалити ‘.h’)

```
// hello1.cpp
#include <cstdio> // Link section: , бібліотека стандартного
          // вводу-виводу C інтегрована як C++ бібліотека
#include <cmath> // заголовочний файл,
         //бібліотека математичних функцій з C в C++
int main() // головна функція (main function): точка входу (entry point)
{
   float x; //визначаємо дійсну (одинарної точності) змінну 'x'
   scanf("%F",&x); // введення змінної 'x'
   double y=sin(x); /* Вираз (expression): виклик функції sin,
              обчислення виразу та
            ініціалізація дійсної змінної (подвійною точності) ‘y’     */
    printf("Result y=%f\n",y); // виведення значення змінної y
}
```

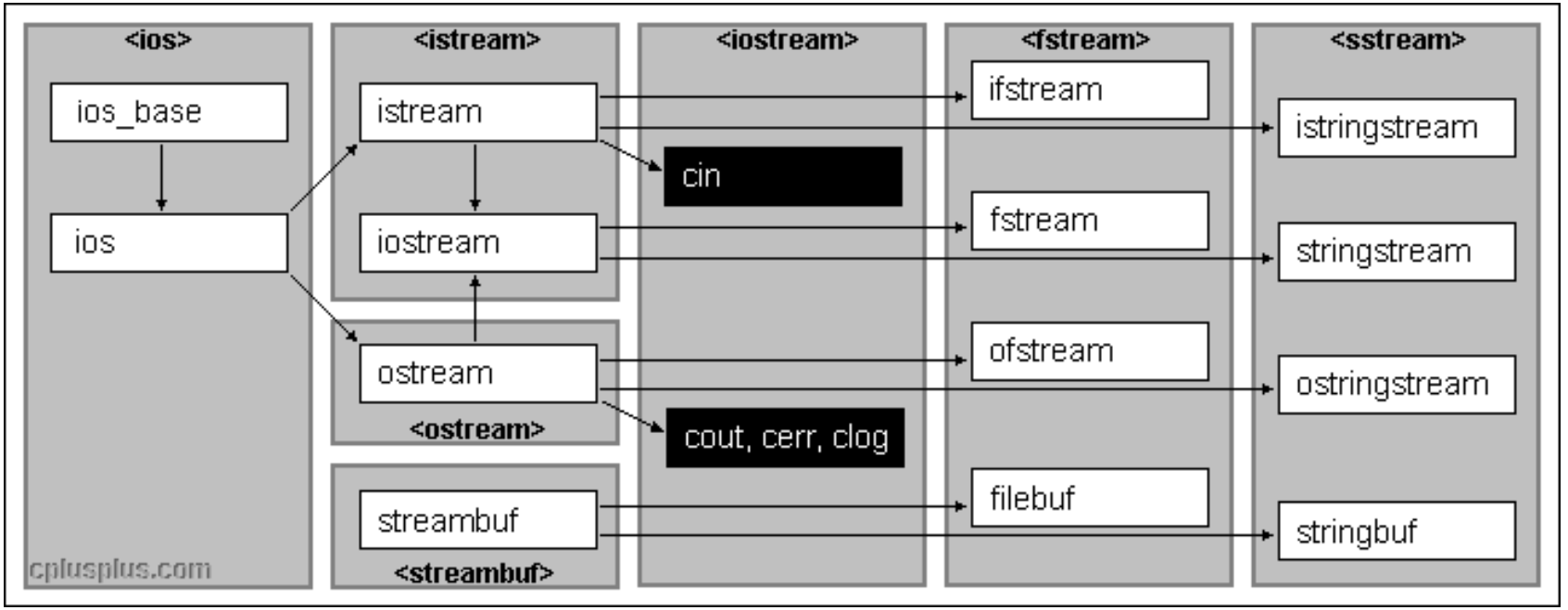
Компіляція: g++ hello1.cpp.

C++ 98 «Hello World» приклади (два варіанти роботи з std)

```
#include <iostream> /* Бібліотека функцій введення-виведення на Сі++ */


int main(){ // точка входу (головна або драйвер функція)
  std::cout<<"Hello\n";   // команда виводу на консоль з простору  std

  std::cout<<"Hello"<<std::endl;   // команда виводу на консоль з простору  std
}
```

```
#include <iostream> /* Бібліотека функцій введення-виведення на Сі++ */

using namespace std; // Вказали простір імен std для всієї програми
int main(){

   cout<<"Hello\n";// команда виводу на консоль
    cout<<"Hello"<<endl;
}
```

Страшна картина класів введення/виведення в C++



cin - екземпляр класу стандартного введення
cout — екземпляр класу стандартного виведення
cerr — екземпляр класу стандартного виводу повідомлень про помилку
clog - екземпляр класу стандартного виводу повідомлень про помилку(буферизований)

Оператор визначення меж :: (scope resolution operator)

Оператор визначення меж **::** служить для керування межею та областю доступу для змінних та функцій.

Зокрема ним можна вказати клас та простір імен звідки береться змінна або функція.

Тобто **std::cout - з std** беремо об'єкт **cout**

Приклад.

```
class X { ****
      static int countX ;
}
int X::countX = 10;
```

using namespace Простір;
застосовує операторну дію Простір:: до всіх класів файлу
Наприклад,
using namespace std;
створює std::cout, std::cin, std::xxx замість cout, cin, xxx

Потоковий ввод з консолі

```
using namespace std;
cin [>> values];
або
std::cin [>> values];
int x, y, z;
double a,b,c;
char ch;
char s[10];
cin>>x;
cin>>a;
cin>>s;
cin >> x >> y;
cin>>z>>ch>>s>>b;
```

Застосовує оператор **>>** з класу **cin** до **values**

Зчитує з потоку **cin** у **value**:
cin >>value1 **>>** value2

1) Введення декількох розділяється пропуском (« ») або **Enter** (дані типу **char** розділяти пропуском необов'язково);
2) потік введення ігнорує пропуски;
3) операція припиняється, коли всі змінні одержують значення.

Потокове виведення з консолі

```
using namespace std;
cout [<< values];
```

або

```
std::cout [<< values];
```

Застосовує оператор **<<** з класу cin до **values**

```
int x, y, z;
double a,b,c;
char ch;
char s[10];
```

Зчитує з **value** в поток **cout**:
cout << value1 << value2

```
cout<<x;
cout<<a<<"\n";
cout<<s<<endl;
cout<<x <<","<<y<<";";
cout<<"Значення\t z="<<z<<ch<<s;
```

Приклад введення/виведення на Ci++

```
#include <iostream> // об’єкти потокового вводу/виводу cin/ cout
#include <cmath> // математичні функції sqrt, cos, log

int main(){    // головна програма
   float C=1.231f; // ініціалізували дійсну змінну
   float x,y; // задекларували 2 дійсні змінні
   std::cout<<"x="; // виводимо підказку користувачу
   std::cin>>x; // вводимо x
   std::cout<<"y="; // виводимо підказку користувачу
   std::cin>>y; // вводимо y
   double A; // задекларували змінну для результату
   if(x<=y) {
      A = x * y - C * y * sqrt(y); // обчислення
   } else {
      A = cos(x) + log(y);
   }
std::cout<<"A="<<A<<std::endl; // виводимо результат, стаємо на новий рядок
std::cin.get(); //те само getchar();  чекаємо вводу символу для затримки
}
```

Опції виведення: setf(long mask)/unsetf(long mask)

dec - десяткова система числення (С.Ч.)
hex - шістнадцяткова С.Ч.
oct - вісімкова С.Ч.
scientific - числа з плаваючою крапкою(n.xxxEyy)
showbase - виводиться основа С.Ч. у виді префікса
showpos - при виводі позитивних числових значень виводиться знак плюс
uppercase - замінює нижній регістр на верхній (м - М)
left - дані вирівнюються по лівому краю поля виводу
right - по правому
internal - додаються символи заповнювачі між усіма цифрами
skipws - знаки пробілу,табуляції і переходу на новий рядок відкидаються

```
std::cout.setf(std::ios_base::hex | std::ios::basefield);
cout.setf(ios::hex | ios::uppercase |ios::showbase); //можемо комбінувати
cout.setf(ios::showpos); // насправді вклаючаємо std::ios
cout << "d = " << d << "";
cout.setf(ios::dec | ios::right| ios::scientific);
cout.unsetf(ios::hex);
cout << "n = " << n << "";
```

```
// приклад модифікації виводу
#include <iostream>     // std::cout, std::dec, std::hex, std::oct

int main () {
  int n = 70;
  std::cout << std::dec << n << '\n';  // вивели в десятковій СЧ
  std::cout << std::hex << n << '\n'; // вивели в шістнадцятковій СЧ
  std::cout << std::oct << n << '\n'; // вивели в вісьмеричній СЧ

 std::cout.flags ( std::ios::right | std::ios::hex | std::ios::showbase );
 std::cout.width (10); // ширина виводу - 10
 std::cout << 100 << '\n';  // вирівнювання зправа 16 СЧ показує її – xxx0x124;

  return 0;
}
/*
70
46
106 */
```

Встановлення точності виведення

```
#include <iostream>
#include <cmath>
int main() {
double x;
std::cout.precision(4); // точність виводу 4 знаки
std::cout.fill('0');  // решту заповнити 0-ми
std::cout << " x / Корінь(x)"<<std::endl;
for (x = 1.0; x <= 6.0; x++) {
  std::cout.width(7); // ширина виводу - 7
  std::cout << x << " ";
  std::cout.width(7);
  std::cout << sqrt(x) << " \n";
}
 }
```

Модуль <iomanip>

endl - новий рядок та очищення потоку
flush - видає вміст буфера потоку у пристрій
setbase (int base) - задає основу системис числення для цілих чисел(8,10,16)
setfill (int c) - встановлює символ-заповнювач
setprecision(int n) - встановлює точність чисел з плаваючою крапкою
setw (int n) - встановлює мінімальну ширину поля виводу
setiosflags(iosbase::long mask) - встановлює ios-прапорці згідно з mask

```
int main() {
double x = 45.12345;
std::cout << "x = " << std::setprecision(4) << std::setfill('0') << std::setw(7) << x << std::endl;

std::cout<<std::setiosflags(std::ios::uppercase)<<"hello"<<
                                                    std::setiosflags(std::ios::floathex)<< x;

std::cout << 456 << 789 << 123 << std::endl;
std::cout << std::setw(5) << 456 << std::setw(5) << 789 << std::setw(5) << 123 << std::endl;
std::cout << std::setw(7) << 456 << std::setw(7) << 789 << std::setw(7) << 123 << std::endl;
}
```

Деякі функції(методи) cin та cout

// Читання даних
getline (char* s, streamsize n, char delim=EOF) // читає рядок з вхідного потоку;
get () // читає символ з вхідного потоку;
ignore (streamsize n = 1, int delim = EOF)) // пропускає вказану кількість елементів від поточної позиції;
read (char* s, streamsize n) // читає вказану кількість символів з вхідного потоку і зберігає їх в буфері (неформатований ввод).
// Запис даних
flush () // очищує вміст буфера в файл (при буферізованому вводе-виводі);
put (char c) // виводить символ в поток;
write (const char* s, streamsize n) // виводить в потік вказану кількість символів з буферу (неформатоване виведення).

Приклад застосування функцій вводу

```
#include <iostream>

int main(void) {
const size_t len = 100;
char name[len];
int count = 0;
std::cout << "Enter your name" << std::endl;
std::cin.getline(name, len); /*  getline () дозволяє безпечно прочитати рядок name
з роздільниками, максимальна довжина рядку len */
count = std::cin.gcount(); // реальна кількість введених символів
/* Зменшуємо значення лічильника на 1, тому що getline() не
поміщає обмежувач в буфер*/
std::cout << "Number of symbols is " << count - 1 << std::endl;
std::cin.get(); // затримка до натиснення символу

std::cin.ignore(256,' '); // ігноруємо до 256 символів поки не введемо роздільник
std::cin>>count;  // а потім зчитаємо число
std::cout<<count; // та виведемо його
}
```

Файлові потоки

ofstream (<ofstream> або <fstream>) для створення та запису у файл
ifstream (<ifstream> або <fstream>) для читання з файлу
fstream (<fstream>) для читання та запису

```
// Для виводу
ofstream outfile;
outfile.open("File.txt");
//або
ofstream outfile("File.txt");
//або
fstream outfile("File.txt ",ios::out);
```

```
// Для вводу
ifstream infile;
infile.open("File.txt");
//або
ifstream infile("File.txt");
//або
fstream infile("File.txt ",ios::in);
```

Прапорці режимів відкриття файлу

No	Прапори режимів роботи з файлами
1	ios::app - режим додавання (Append mode). Введення додається в кінець файлу
2	ios::ate — відкриває файл для виводу та ставить маркер до кінця файлу
3	ios::in — відкриває файл для читання.
4	ios::out — відкриває файл для запису.
5	ios::trunc — якщо файл існує його вміст буде збережений.

```
ofstream outfile;
outfile.open("file.dat", ios::out | ios::trunc );

fstream  afile;
afile.open("file.dat", ios::out | ios::in );
```

Приклад роботи з файлом

```
// Відкриваємо файл для виводу
ofstream ofile("Test.txt");

// if (ofile.bad()) – те саме
if (!ofile) {
cout << "Файл не відкритий. ";
return -1;
}
ofile << "Hello!" << n;
// Закриваємо файл
ofile.close();
```

```
// Відкриваємо той же файл для вводу
ifstream ifile("Test.txt");

// if (ifile.bad()) – те саме
if ( !ifile ) {
cout << "Файл не відкритий.";
return -1;
}
char str[80];
ifile >> str >> n;
cout << str << " " << n << endl;
ifile.close(); // Закриваємо файл
```

```
#include <iostream> // стандартне введення-виведення
#include <fstream>  // функції роботи з файлами
using namespace std;
const char * filename = "testfile2.txt";
int main () { // створення потоку, відкриття файлу для записів в текстовому режимі,
 ofstream ostr; // файлова змінна (файл для запису)
 ostr.open (filename); // відкрити файл
 if (ostr) { // якщо він ОК
  for(int i = 0; i <16; i ++) {
    ostr << i * i << endl; // записати туди квадрати чисел
    if (ostr.bad ()) { // перевірка роботи
      cerr<<"Невиправна помилка запису”<< endl; //виводимо в консоль для помилок
      return 1;
    }
  }
  ostr.close (); // не забули закрити файл
 }
 else{  // якщо помилка відкриття віводимо в консоль для виводу помилок
    cerr<< "Вихідний файл відкриває помилку"<< filename <<"\t";
   return 1;
}
```

```
int data; // читання даних, форматоване виведення на консоль, закриття файлу.
 int counter = 0;
 ifstream istr (filename); // файл для читання
 if (istr) { // якщо він ОК
   while (! (istr >> data).eof ()) { // доки не кінець файлу читаємо
      if (istr.bad ()) { // перевіряємо всі читання
          cerr << "Невиправна помилка читання" << endl;
          return 2;
      }
      cout.width (8);  // виведення по 8 символів кожен
      cout << data;
      if (++ counter% 4 == 0) { cout << endl;} // по 4 штуки числа в ряд
   }
   istr.close (); // не забули закрити файл
  }
  else{ // якщо не ОК – вивели цю інфу в поток помилок
    cerr << "Помилка відкриття вхідного файлу"<< filename <<"\t";
    return 2;
  }
  return 0;
}
```

Булевий тип

```
#include <iostream>     // std::cout, std::boolalpha, std::noboolalpha

int main () {
  bool a = false;
  bool b = true;
  std::cout << std::boolalpha << b && a << '\n';
  std::cout << std::noboolalpha << b || a << '\n';
  return 0;
}
/*
Результат:
false
1
*/
```

Перезавантаження функцій

```
int add(int x, int y){  // версія 1
        return x + y;
}
double add(double x, double y){ // версія 2
        return x + y;
}
int add(int x, int y, int z){ // версія 2
    return x + y + z;
}
int main(){
     int z = add(2,1);
     double z1 = add(2.0,1);
     z = add(1,2,1);
}
```

Посилання

```
int i = 3; // змінна
int *pointer_i = &i; //вказівник вказує на адресу цієї змінної
int &ref_i = i; // адреса цієї змінної

cout<<i<<","<<*pointer_i<<","<<ref_i<<"\n"; // 3, 3, 3

*pointer_i  =  4; // зміна значення за адресою
ref_i = 4; // зміна значення за адресою

cout<<i<<","<<*pointer_i<<","<<ref_i<<"\n"; // 4, 4, 4

void swap(int &v1, int &v2){ // функція замін місцями змінних
        int temp = v1;
        v1 = v2;
        v2 = temp;
        }
```

```
 int *ptr=0, x=9; // ініціалізація змінних та вказівника
ptr=&x; // вказівник на змінну "x"
int & j=x; // посилання; посилання на змінну "x"

 std::cout << "x=" << x << std::endl;
std::cout << "&x=" << &x << std::endl;

std::cout << "j=" << j << std::endl;
std::cout << "&j=" << &j << std::endl;

std::cout << "*ptr=" << *ptr << std::endl;
std::cout << "ptr=" << ptr << std::endl;
```

Результат:

x=9
&x=0x7ffe137f6414
j=9
&j=0x7ffe137f6414
***ptr=9**
ptr=0x7ffe137f6414

```
void fn1(std::string s);
void fn2(const std::string& s);
void fn3(std::string& s);
void fn4(std::string* s);

void bar() {
   std::string x;
   fn1(x);  // можна модифікувати x
   fn2(x);  // не можна модифікувати x (без const_cast)
   fn3(x);  // можна модифікувати x!
   fn4(&x); // можна модифікувати x  (і каже про це)
}
```

Виділення памяті за допомогою new// та видалення за допомогою delete

```
int *p1 = NULL; // задекларували вказівник та ініціалізували в 0
p1 = new int; // виділили пам’ять під нього

int *p2 = new int; // задекларували та виділили

int *p3 = new int(25); // ініціалізували вказівник конкретними значеннями за адресами
float *p4 = new float(75.25);

cout<<p1<<","<<p2<<","<<p3<<","<<p4<<endl;
 //0x559c8763ee70,0x559c8763ee90,0x559c8763eeb0,0x559c8763eed
cout<<*p1<<","<<*p2<<","<<*p3<<","<<*p4<<endl;
// 0,0,25,75.25
delete p1;
delete p2;
delete p3;
delete p4;
```

Результат:

0x559c8763ee70,0x559c8763ee90,0x559c8763eeb0,0x559c8763eed0
0,0,25,75.25

Виділення памяті під динамічний масив

```
int *p5 = new int[10]; // масив з 10 цілих чисел
int *p6 = new int[5] {10,2,3,4,}; // ініціалізуємо масив з 5 чисел

int  m =10;
int *p7 = new int[*p6]; // масив з  ? цілих чисел
int *p8 = new int[m]; // динамічний масив  з m чисел

p7[9]=1; p7[0]=2;

cout<<p5<<","<<p6<<endl; // 0x55838062f300,0x55838062eed0
cout<<p5[1]<<","<<p6[1]<<";"<<p7[0]<<”,”<<p8[9]<<endl; // 0,1;2;0

delete[] p5;
delete[] p6;
delete[] p8;
```

Результат:
0x55838062f300,0x55838062eed0
0,1;2;0

Різниця між **new** та malloc

new	malloc
Викликає конструктор	Не викликає конструктор
Оператор	Функція
Повертає змінну відповідного типу	Повертає void*
При невдачі повертає виключення std :: bad_alloc	При невдачі повертає NULL
Може бути перевантаженим	Не може бути перевантаженим
Розмір рахується компілятором	Розмір рахується програмістом