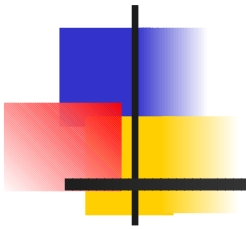
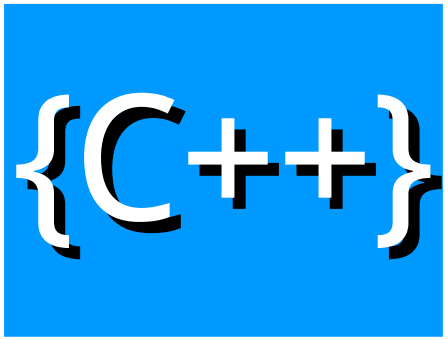
Random-Access Files



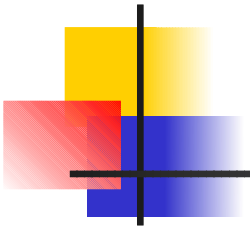
Week 12



Yang-Cheng Chang
Yuan-Ze University
yczhang@saturn.yzu.edu.tw

Assignment 12

- Write a program to convert data format.
  - —Read data from a text file(records.csv).
  - —Write these data to a binary file(records.dat).
  - —There is a additional field 'bmi' in the data format of our binary file. You must calculate and assign the value of 'bmi' when convert data to our binary file.
- Read data which its field 'bmi' is belong to category 'underweight' from the binary file(records.dat). And show these data.



The format of input file(records.csv)

id,name,weight,height

1,victor,74,161
2,may,45,153
3,joe,76,164

The format of output file(records.dat)

```
struct record {
    int id;
    unsigned char name[16];
    int weight;
    int height;
    float bmi;
};
```

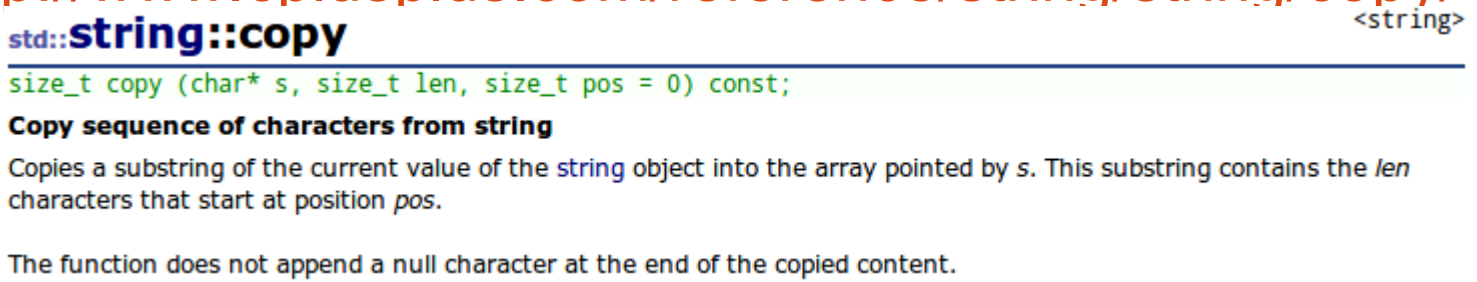
BMI

$$BMI = \frac{mass(kg)}{height(m)^2}$$

Category	BMI range(kg/m^2)
Underweight	under 18.5
Normal	from 18.5 to 24
Overweight	from 24 to 27
Moderately obese	from 27 to 30
Severely obese	from 30 to 35
Very severely obese	over 35

std::string::copy – unsafe way

Reference:
http://www.cplusplus.com/reference/string/string/copy/



std::string::copy <string>

size_t copy (char* s, size_t len, size_t pos = 0) const;

Copy sequence of characters from string

Copies a substring of the current value of the string object into the array pointed by *s*. This substring contains the *len* characters that start at position *pos*.

The function does not append a null character at the end of the copied content.

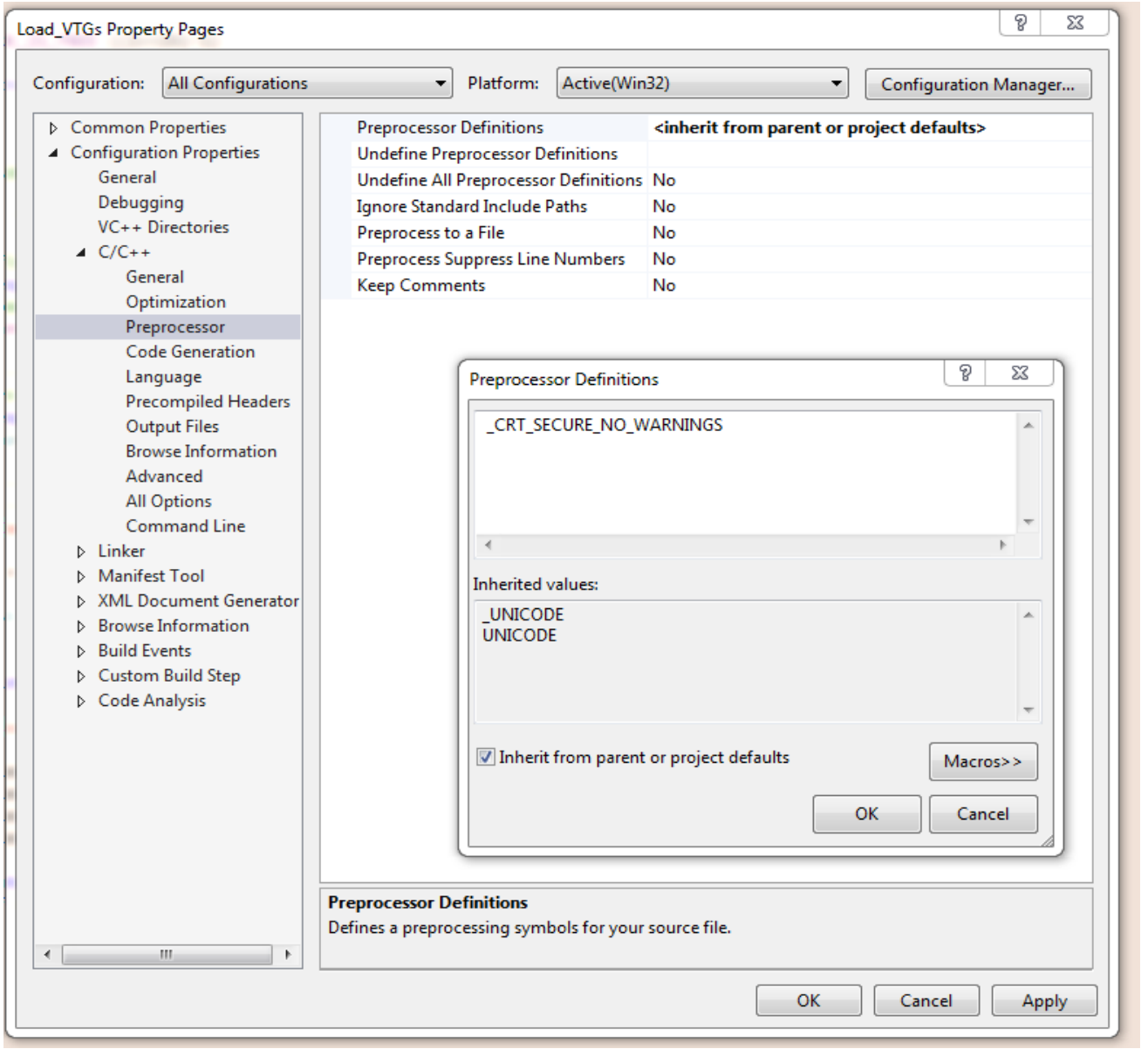
```
// string::copy
#include <iostream>
#include <string>

int main ()
{
  char buffer[20];
  std::string str ("Test string...");
  std::size_t length = str.copy(buffer,6,5); //
  buffer[length]='\0';
  std::cout << "buffer contains: " << buffer << '\n';
  return 0;
}
```

WARNING: C4996

- Some functions are unsafe, and deprecated
- Solution
  - —Use the alternative secure function
    http://msdn.microsoft.com/zh-cn/library/wd3wzwts%28v=vs.110%29.aspx
  - —Ignore WARNING
    - Project Name → Alt-ENTER
    - Configurations:All Configurations
    - Click on the Preprocessor Definitions line to invoke its editor
    - Choose Edit...
    - Copy "_SCL_SECURE_NO_WARNINGS" into the Preprocessor Definitions white box on the top.

IGNORE WARING C4996



Use std::copy – The C++ Way

```
// Use std::copy to convert std::string to char*

#include <iostream>
#include <string>

int main ()
{
    std::string str ("Test string...");
    char * writable = new char[str.size() + 1];
    std::copy(str.begin(), str.end(), writable);
    writable[str.size()] = '\0'; // don't forget the terminating 0
    std::cout << "buffer contains: " << writable << '\n';
    // don't forget to free the string after finished using it
    delete[] writable;
    return 0;
}
```

Access to a binary file

- The prototype of the write function

  ostream& write(const char *s, streamsize n);

std::ostream::write <ostream> <iostream>

```
ostream& write (const char* s, streamsize n);
```

Write block of data

Inserts the first *n* characters of the array pointed by *s* into the stream.

This function simply copies a block of data, without checking its contents: The array may contain *null characters*, which are also copied without stopping the copying process.

- The prototype of the read function

  istream& read(char *s, streamsize n);

std::istream::read <istream> <iostream>

```
istream& read (char* s, streamsize n);
```

Read block of data

Extracts *n* characters from the stream and stores them in the array pointed to by *s*.

This function simply copies a block of data, without checking its contents nor appending a *null character* at the end.

The Example of ostream::write

```
struct Person
{
    char name[50];
    int age;
    char phone[24];
};
int main()
{
    Person me = {"Robert", 28, "364-2534"};
    Person book[30];
    int x = 123;
    double fx = 34.54;
    ofstream outfile;
    outfile.open("junk.dat", ios::binary | ios::out);
    outfile.write((char*)&x, sizeof(int)); // sizeof can take a type
    outfile.write((char*)&fx, sizeof(fx)); // or it can take a variable name
    outfile.write((char*)&me, sizeof(me));
    outfile.write((char*)book, 30*sizeof(Person));
    outfile.close();

}
```

The Example of istream::read

```
struct Person
{
    char name[50];
    int age;
    char phone[24];
};

int main()
{
    Person list[] = {
                    {"Robert", 28, "364-2534"}
                   ,{"Tim",    30, "364-7777"}
                   ,{"Bob",    21, "364-4538"}
                };
    ofstream outfile;
    outfile.open("junk.dat", ios::binary | ios::out);
    outfile.write((char*)list, 3*sizeof(Person));
    outfile.close();

    ifstream infile;
    infile.open("junk.dat", ios::binary | ios::in);
    while(!infile.eof()){
        Person tmp;
        infile.read((char *)&tmp, sizeof(Person));
        if (!infile.eof()){
            cout << setw(10) << tmp.name << setw(10) << tmp.age << setw(10) << tmp.phone << endl;
        }
    }
}
```