Pointers

Ho Dac Hung

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointers

- Pointers are an important feature of C++ (and C), while many other languages, such as Visual Basic and Java, have no pointers at all. However, in some situations pointers provide an essential tool for increasing the power of C++.

cuu duong than cong . com

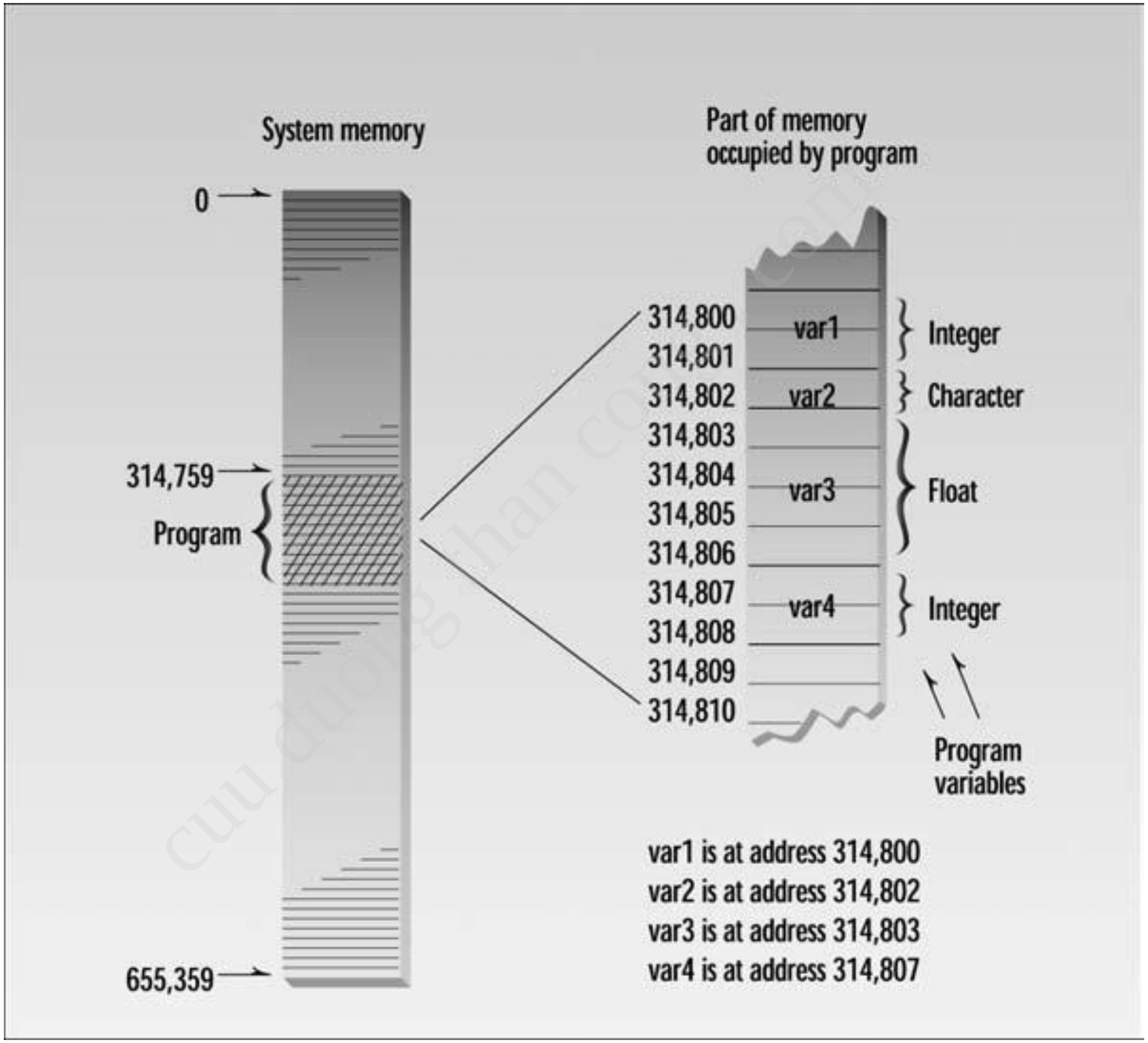
CuuDuongThanCong.com https://fb.com/tailieudientucntt

Addresses and Pointers

- The ideas behind pointers are not complicated. Here's the first key concept: Every byte in the computer's memory has an address. Addresses are numbers, just as they are for houses on a street. The numbers start at 0 and go up from there—1, 2, 3, and so on. If you have 1MB of memory, the highest address is 1,048,575.
- Your program, when it is loaded into memory, occupies a certain range of these addresses. That means that every variable and every function in your program starts at a particular address.

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt



CuuDuongThanCong.com https://fb.com/tailieudientucntt

The Address-of Operator &

- You can find the address occupied by a variable by using the address-of operator &.

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

The Address-of Operator &

```
int main()
{
    int var1 = 11; //define and initialize
    int var2 = 22; //three variables
    int var3 = 33;
    cout << &var1 << endl //print the addresses
    << &var2 << endl //of these variables
    << &var3 << endl;
    return 0;
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointer Variables

- The potential for increasing our programming power requires an additional idea: variables that hold address values. We’ve seen variable types that store characters, integers, floating-point numbers, and so on. Addresses are stored similarly. A variable that holds an address value is called a pointer variable, or simply a pointer.

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
int main()
{
    int var1 = 11; //two integer variables
    int var2 = 22;
    cout << &var1 << endl //print addresses of variables
    << &var2 << endl << endl;
    int* ptr; //pointer to integers
    ptr = &var1; //pointer points to var1
    cout << ptr << endl; //print pointer value
    ptr = &var2; //pointer points to var2
    cout << ptr << endl; //print pointer value
    return 0;
}
```

cuu duong than cong. com

Accessing the Variable Pointed To

- Suppose that we don't know the name of a variable but we do know its address. Can we access the contents of the variable? There is a special syntax to access the value of a variable using its address instead of its name.

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
int main()
{
    int var1 = 11; //two integer variables
    int var2 = 22;
    int* ptr; //pointer to integers
    ptr = &var1; //pointer points to var1
    cout << *ptr << endl; //print contents of pointer (11)
    ptr = &var2; //pointer points to var2
    cout << *ptr << endl; //print contents of pointer (22)
    return 0;
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointer to void

- Before we go on to see pointers at work, we should note one peculiarity of pointer data types. Ordinarily, the address that you put in a pointer must be the same type as the pointer.
- However, there is an exception to this. There is a sort of general-purpose pointer that can point to any data type. This is called a pointer to void, and is defined like this:

```
        void* ptr;
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
int main()
{
    int intvar; //integer variable
    float flovar; //float variable
    int* ptrint; //define pointer to int
    float* ptrflo; //define pointer to float
    void* ptrvoid; //define pointer to void
    // ptrint = &flovar; //error, float* to int*
    // ptrflo = &intvar; //error, int* to float*
    ptrvoid = &intvar; //ok, int* to void*
    ptrvoid = &flovar; //ok, float* to void*
    return 0;
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointer to void

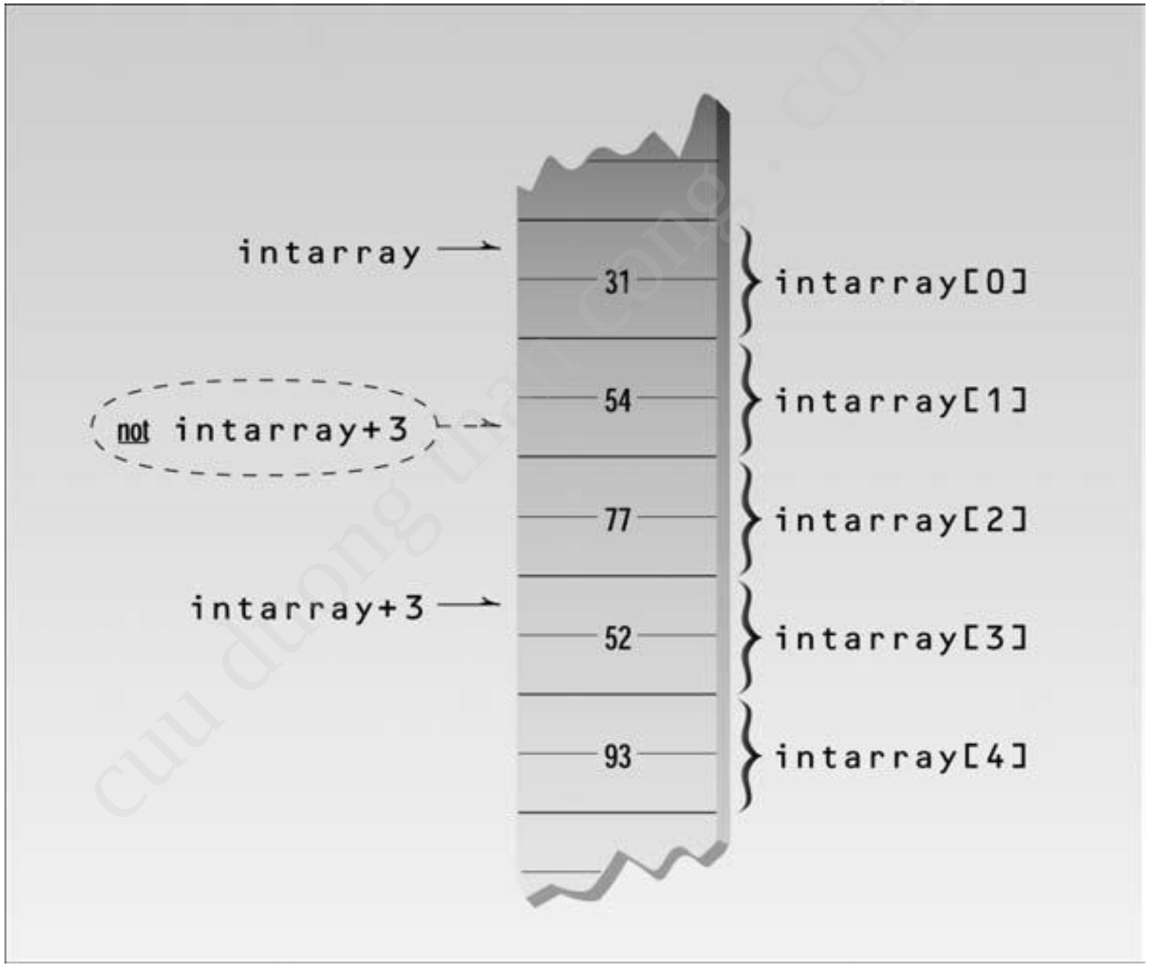
- If for some unusual reason you really need to assign one kind of pointer type to another, you can use the reinterpret_cast.

```
ptrint = reinterpret_cast<int*>(flovar);
ptrflo = reinterpret_cast<float*>(intvar);
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointers and Arrays



cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointers and Arrays

```
int main()
{ //array
    int intarray[5] = { 31, 54, 77, 52, 93 };
    for(int j=0; j<5; j++) //for each element,
        cout << intarray[j] << endl; //print value
    return 0;
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointers and Arrays

```
int main()
{ //array
    int intarray[5] = { 31, 54, 77, 52, 93 };
    for(int j=0; j<5; j++) //for each element,
        cout << *(intarray+j) << endl; //print value
    return 0;
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointer Constants and Pointer Variables

- Suppose that, instead of adding j to intarray to step through the array addresses, you wanted to use the increment operator. Could you write *(intarray++)?
- The answer is no, and the reason is that you can't increment a constant The expression intarray is the address where the system has chosen to place your array, and it will stay at this address until the program terminates.
- But while you can't increment an address, you can increment a pointer that holds an address.

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointer Constants and Pointer Variables

```
int main()
{
    int intarray[] = { 31, 54, 77, 52, 93 }; //array
    int* ptrint; //pointer to int
    ptrint = intarray; //points to intarray
    for(int j=0; j<5; j++) //for each element,
        cout << *(ptrint++) << endl; //print value
    return 0;
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointers and Functions

- By value
- By reference
- By pointer

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
int main()
{
    void centimize(double&); //prototype
    double var = 10.0; //var has value of 10 inches
    cout << “var = “ << var << “ inches” << endl;
    centimize(var); //change var to centimeters
    cout << “var = “ << var << “ centimeters” << endl;
    return 0;
}
void centimize(double& v)
{
    v *= 2.54; //v is the same as var
}
```

cuuduong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
int main()
{
    void centimize(double*); //prototype
    double var = 10.0; //var has value of 10 inches
    cout << “var = “ << var << “ inches” << endl;
    centimize(&var); //change var to centimeters
    cout << “var = “ << var << “ centimeters” << endl;
    return 0;
}
void centimize(double* ptrd)
{
    *ptrd *= 2.54; //*ptrd is the same as var
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
int main()
{
    void centimize(double*); //prototype
    double varray[MAX] = { 10.0, 43.1, 95.9, 59.7, 87.3 };
    centimize(varray); //change elements of varray to cm
    for(int j=0; j<MAX; j++) //display new array values
        cout << “varray[“ << j << “]=”
        << varray[j] << “ centimeters” << endl;
    return 0;
}
void centimize(double* ptrd)
{
    for(int j=0; j<MAX; j++)
        *ptrd++ *= 2.54; //ptrd points to elements of varray
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Sorting Array Elements

```
void order(int* numb1, int* numb2)
{
    if(*numb1 > *numb2) //if 1st larger than 2nd,
    {
        int temp = *numb1; //swap them
        *numb1 = *numb2;
        *numb2 = temp;
    }
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Sorting Array Elements

```
void bsort(int* ptr, int n)
{
    void order(int*, int*); //prototype
    int j, k; //indexes to array
    for(j=0; j<n-1; j++)
        for(k=j+1; k<n; k++)
            order(ptr+j, ptr+k);
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
int main()
{
    void bsort(int*, int); //prototype
    const int N = 10; //array size
    //test array
    int arr[N] = { 37, 84, 62, 91, 11, 65, 57, 28, 19, 49 };
    bsort(arr, N); //sort the array
    for(int j=0; j<N; j++) //print out sorted array
        cout << arr[j] << " ";
    cout << endl;
return 0;
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointers to String Constants

```
int main()
{
    char str1[] = “Defined as an array”;
    char* str2 = “Defined as a pointer”;
    cout << str1 << endl; // display both strings
    cout << str2 << endl;
    // str1++; // can’t do this; str1 is a constant
    str2++; // this is OK, str2 is a pointer
    cout << str2 << endl; // now str2 starts “efined...”
    return 0;
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

The const Modifier and Pointers

- The use of the const modifier with pointer declarations can be confusing, because it can mean one of two things, depending on where it’s placed.

const int* cptrInt; //cptrInt is a pointer to constant int

int* const ptrcInt; //ptrcInt is a constant pointer to int

cuu duong than cong . com

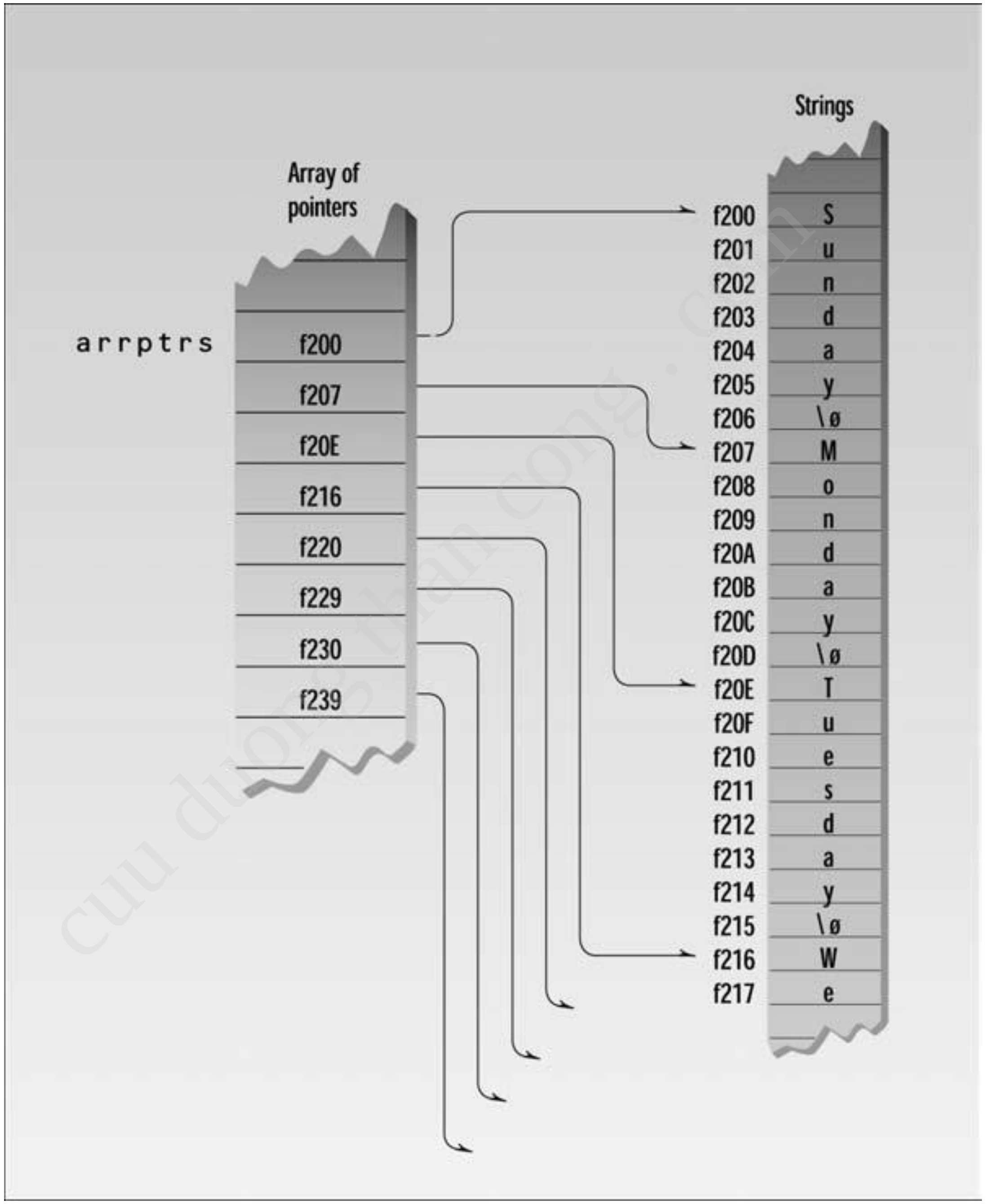
CuuDuongThanCong.com https://fb.com/tailieudientucntt

Arrays of Pointers to Strings

```
int main()
{ //array of pointers to char
    char* arrptrs[DAYS] = { “Sunday”, “Monday”,
        “Tuesday”, “Wednesday”, “Thursday”,
        “Friday”, “Saturday” };
    for(int j=0; j<DAYS; j++) //display every string
        cout << arrptrs[j] << endl;
    return 0;
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt



CuuDuongThanCong.com https://fb.com/tailieudientucntt

Memory Management: new and delete

- C++ provides a different approach to obtaining blocks of memory: the new operator. This versatile operator obtains memory from the operating system and returns a pointer to its starting point.

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Memory Management: new and delete

- If your program reserves many chunks of memory using new, eventually all the available memory will be reserved and the system will crash. To ensure safe and efficient use of memory, the new operator is matched by a corresponding delete operator that returns memory to the operating system.

cuu duong than cong . com

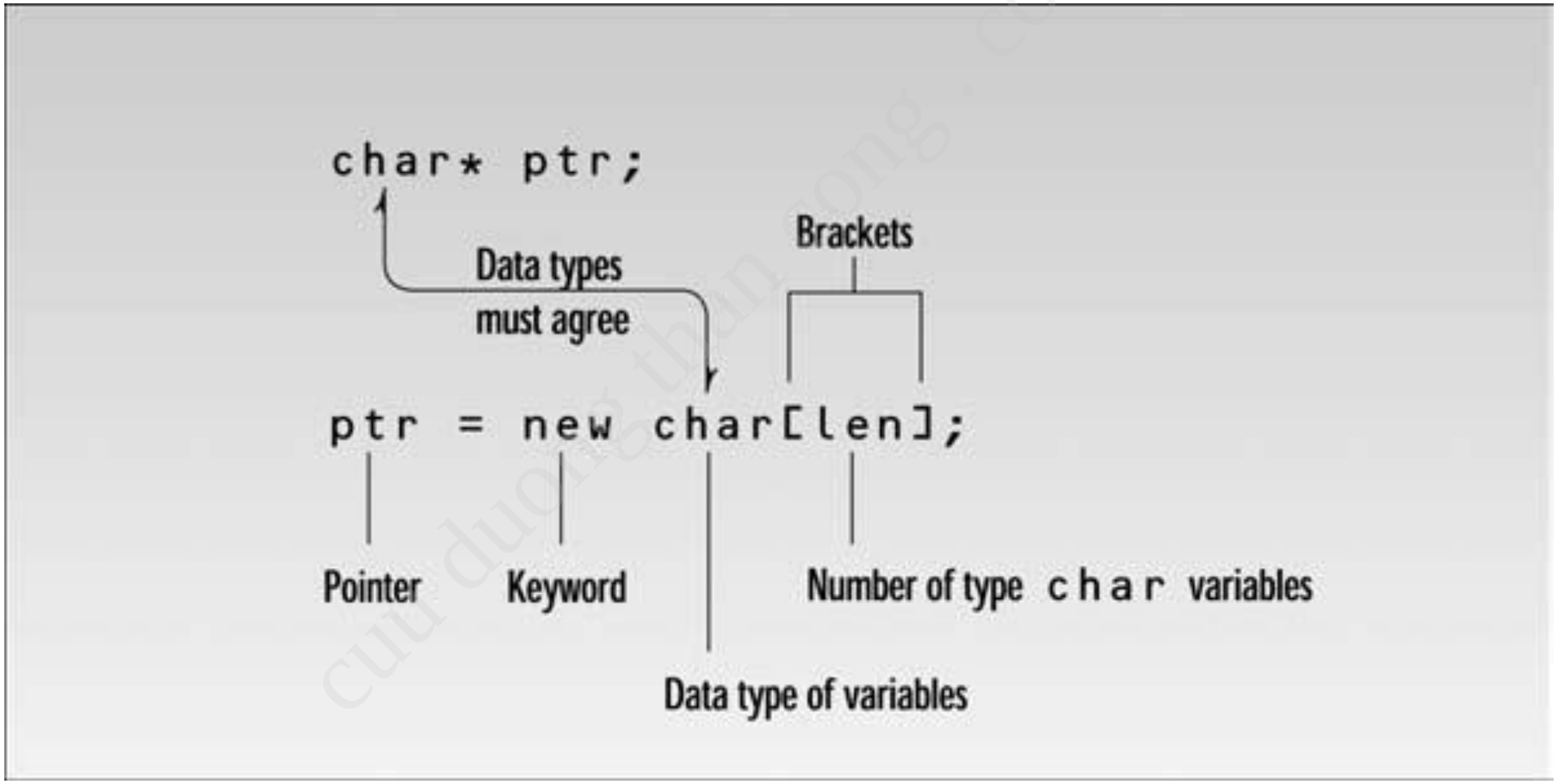
CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
int main()
{
    char* str = “Idle hands are the devil’s workshop.”;
    int len = strlen(str); //get length of str
    char* ptr; //make a pointer to char
    ptr = new char[len+1]; //set aside memory: string +
          ‘\0’
    strcpy(ptr, str); //copy str to new memory area ptr
    cout << “ptr=” << ptr << endl; //show that ptr is now in
          str
    delete[] ptr; //release ptr’s memory
    return 0;
}
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Memory Management: new and delete



cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Pointers to Objects

- Pointers can point to objects as well as to simple data types and arrays.

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
class Distance //English Distance class
{
    private:
        int feet;
        float inches;
    public:
        void getdist() //get length from user
        {
            cout << "\nEnter feet: "; cin >> feet;
            cout << "Enter inches: "; cin >> inches;
        }
        void showdist() //display distance
        { cout << feet << "\'-" << inches << '\"'; }
};
```

cuu duong than cong . com

```
int main()
{
    Distance dist; //define a named Distance object
    dist.getdist(); //access object members
    dist.showdist(); // with dot operator
    Distance* distptr; //pointer to Distance
    distptr = new Distance; //points to new Distance
        object
    distptr->getdist(); //access object members
    distptr->showdist(); // with -> operator
    cout << endl;
    return 0;
}
```

cuuduong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

Referring to Members

distptr.getdist(); // won’t work; distptr is not a variable

(*distptr).getdist(); // ok but inelegant

distptr->getdist(); // better approach

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

An Array of Pointers to Objects

- A common programming construction is an array of pointers to objects. This arrangement allows easy access to a group of objects, and is more flexible than placing the objects themselves in an array.

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
class person //class of persons
{
    protected:
        char name[40]; //person’s name
    public:
        void setName() //set the name
        {
            cout << “Enter name: “;
            cin >> name;
        }
        void printName() //get the name
        {
            cout << “\n Name is: “ << name;
        }
};
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
int main()
{
    person* persPtr[100]; //array of pointers to persons
    int n = 0; //number of persons in array
    char choice;
    do //put persons in array
    {
         persPtr[n] = new person; //make new object
        persPtr[n]->setName(); //set person’s name
        n++; //count new person
        cout << “Enter another (y/n)? “; //enter another
        cin >> choice; //person?
    }
    while( choice==’y’ );
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt

```
    for(int j=0; j<n; j++) //print names of
    { //all persons
        cout << “\nPerson number “ << j+1;
        persPtr[j]->printName();
    }
    cout << endl;
    return 0;
} //end main()
```

cuu duong than cong . com

CuuDuongThanCong.com https://fb.com/tailieudientucntt